Abstract

Title of Dissertation: Fast Evaluation and Dynamic Control of Integrated Services Networks

Ibrahim Matta, Doctor of Philosophy, 1995

Dissertation directed by: Associate Professor Udaya Shankar
Department of Computer Science

Integrated services networks, such as ATM (Asynchronous Transfer Mode) networks, are expected to operate at gigabit per second rates and provide various virtual-circuit and datagram services. For this purpose, new control algorithms (e.g. scheduling, admission, routing) have been proposed. The algorithms are often adaptive, resulting in complex time-dependent interactions. This renders traditional evaluation tools ineffective; analytical approaches are typically too coarse, and simulation approaches are often too expensive. The goal of our research is to develop accurate analytical models that account for the interaction and time-dependent nature of the control algorithms, while at the same time being inexpensive or easy to solve. This would allow the rapid and tractable evaluation of different design alternatives.

In this dissertation, we develop both dynamic models and quasi-static models of integrated networks. Dynamic models can be used to evaluate both virtual-circuit and datagram services. We solve dynamic models using a new iterative method, referred to as the *Z-iteration*. Our method is both accurate and fast. It permits the joint evaluation of various scheduling, admission, and routing schemes used in integrated networks. We show results comparing dynamic routing schemes on a network with NSFNET-backbone topology. We also illustrate the applicability of the Z-iteration to other high-performance systems.

Quasi-static models are suitable for evaluating datagram services for which the quasi-static assumption is reasonable. We analyze a quasi-static model of a datagram network offering different classes of service. We apply the Liapunov function method to derive stability conditions for the routes of the different traffic classes. We show how with scheduling support for routing, the routes of the traffic classes can be isolated, thereby improving the overall network performance.

Fast Evaluation and Dynamic Control of Integrated Services Networks

by

Ibrahim Matta

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1995

Advisory Committee:

Associate Professor Udaya Shankar, Chairman/Advisor
Professor Ashok Agrawala
Assistant Professor Richard Gerber
Professor Armand Makowski
Professor Satish Tripathi

Dedication

To my parents, and my wife

Acknowledgements

I would like to express my sincere appreciation to my advisor, Dr. Udaya Shankar, for his experienced guidance, endless encouragement, and tireless enthusiasm throughout the course of my study and research at the University of Maryland. I feel lucky to have had such a great mentor. He always believed in my abilities, which made me think much harder and challenge myself much more. I am very grateful to him for teaching me many skills to be a successful and productive researcher, and especially for having the patience to teach me how to give a good talk and write a good paper.

I wish to express my deep gratitude to Dr. Ashok Agrawala for serving on my proposal and dissertation committees, for helping me in many ways, and for his financial support during my studies. I thank Dr. Rich Gerber, who was on my proposal and dissertation committees, for his insightful feedback and support. I also thank the other members of my dissertation committee, Dr. Satish Tripathi and Dr. Armand Makowski, for their interest in this dissertation and for their feedback. Thanks are also due to Dr. Jim Anderson for serving on my proposal committee, and Dr. Nagi Wakim for the wonderful teaching experience I had in the Computer Science department at Bowie State University.

I especially acknowledge the help and support accorded to me by the staff of my department and by many friends and colleagues. Special thanks are due to Cengiz Alaettinoğlu, Jean Bolot, Klaudia Dussa-Zieger, Debanjan Saha, Pravin Bhagwat, Ibrahim Kamel, Sedat Akyurek, Sarit Mukherjee, Partho Mishra, Daniel Mossé, Ólafur Gudmundsson, Dheeraj Sanghi, Sanjeev Setia, and James da Silva, and also to Dr. Nancy Lindley and Janet Doherty, who have helped me in numerous administrative affairs.

I would like to thank all my professors at Alexandria University in the department of Computer Science for their teaching and support during my undergraduate and the initial years of my graduate studies.

Finally, I must acknowledge the endless support of my family. I especially thank my wonderful wife Dalia, my parents, and my brother, for their love and encouragement. Their sacrifice and patience made this dissertation possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Integrated services packet-switched networks, such as ATM (Asynchronous Transfer Mode) networks [91], are expected to support a wide variety of applications (e.g., multimedia, voice, mail) with heterogeneous quality-of-service (QoS) requirements. To meet these requirements, new algorithms have been proposed for controlling routing, admission, and scheduling. *Routing* provides a selection of routes, based on cost functions associated with the transmission links. Routing can be on a virtual-circuit basis (needed for guaranteed service) or on a datagram basis (suited for best-effort service). *Admission* defines the criteria used to accept or reject a new incoming application, based on the service requested and the resources available. *Scheduling* defines how link resources (bandwidth, buffers, etc.) are allocated among the different services.

The overall end-to-end performance of the network hinges on the algorithms used in the routing, admission, and scheduling components. The algorithms are often *adaptive*, with parameters being varied dynamically according to service class and current or delayed system state information. Arrival and service statistics are often time-dependent. As a result, there is significant interaction among the three components.

The accurate and fast evaluation of such time-dependent systems is critical to their cost-effective design. Existing evaluation methods for these systems are inadequate. *Analytical methods are typically too coarse.* They usually assume steady-state conditions and do not account for adaptive policies and the effect of delayed feedback. Incorporating adaptive time-dependent behavior makes them analytically intractable and computationally expensive to solve numerically due to the large state space. *Simulation approaches are often too expensive.* They can handle realistic detail and dynamic situations, but they are invariably computationally prohibitive, especially for evaluating high-speed networks where the number of scheduled events (packets, connections, etc.) is usually enormous.

The goal of our research is to develop accurate analytical models that exhibit the essential features of integrated networks, while at the same time being inexpensive or easy to solve.

1.1 Contributions

Time-Dependent Evaluation

In this dissertation, we present a numerical-analytical method, referred to as the *Z-iteration*, to evaluate integrated networks rapidly and accurately, taking into account the interaction and time-dependent nature of the control algorithms. The method is applicable to a general time-dependent multiple-class multiple-resource (MCMR) system, where each class of customers requires a particular set of resources. Customers can be packets, connections, etc., and resources can be buffers, transmission capacity, etc. Because the class of a customer can be assigned when the customer arrives, it is straightforward to model state-dependent control policies such as assigning connections to routes with the least load. The numerical foundation of the *Z*-iteration provides a modeling power close to that of simulation at a fraction of the computation expense, typically less expensive by many orders.

The *Z*-iteration solves for *instantaneous* performance measures. It approximates the MCMR system as a collection of multiple-class single-resource (MCSR) systems. It computes $B_c^r(t)$, the instantaneous blocking probability of class c at resource r, together with $N_c^r(t)$, the instantaneous average number of class-c customers waiting or in service at r, and $U_c^r(t)$, the instantaneous average number of class-c customers in service at r.

The *Z*-iteration depends upon the availability of two steady-state results about each MCSR system r assuming that the arrival and service rates $\lambda_c^r(t)$ and $\mu_c^r(t)$ are constants: (1) an expression for the steady-state B_c^r in terms of the steady-state λ_c^r/μ_c^r; and (2) an expression for the steady-state U_c^r in terms of the steady-state N_c^r, from which we readily obtain an expression for the λ_c^r/μ_c^r in terms of the N_c^r and B_c^r. These two steady-state results are available in the literature for a variety of MCSR systems.

The method obtains an approximation to the relationship between the $B_c^r(t)$ and the $N_c^r(t)$ by replacing in the above expressions B_c^r by $B_c^r(t)$, N_c^r by $N_c^r(t)$, and λ_c^r/μ_c^r by an instantaneous quantity $z_c^r(t)$ that we introduce. This yields for every r two "instantaneous" expressions, one for the $B_c^r(t)$ in terms of the $z_c^r(t)$, and one for the $z_c^r(t)$ in terms of the $N_c^r(t)$ and $B_c^r(t)$. Given the $N_c^r(t)$, we iterate over these two expressions until the $B_c^r(t)$ and $z_c^r(t)$ converge. To obtain the time evolution of these measures, we iterate over a third expression defining the $N_c^r(t+\delta)$ in terms of the $N_c^r(t)$, $\lambda_c^r(t)$, $\mu_c^r(t)$, and $B_c^r(t)$, where δ is the time step for computing the instantaneous measures.

We use the *Z*-iteration to study the performance of an integrated network with NSFNET backbone topology, weighted fair-queueing link scheduling [89], admission control based on "effective bandwidth" [44], and various virtual-circuit routing schemes that adapt to delayed state information expressed in terms of link utilizations.

Quasi-Static Evaluation

The Z-iteration can be used to solve dynamic models that capture the general time-varying behavior of integrated networks offering both virtual-circuit and datagram services. More tractable but somewhat restrictive models, referred to as *quasi-static* models, are sometimes appropriate to evaluate datagram (best-effort) services. These models assume that steady-state is reached between two successive routing updates. This is justifiable because packet transmission times are small compared to the routing update interval, assuming static loading and network topology during each update interval. The link costs are iteratively computed from steady-state queueing results, and routes (and hence the system state) are updated accordingly. Because quasi-static models capture less detail, they generally allow faster and more tractable evaluation over dynamic models. However quasi-static models do not seem appropriate to evaluate virtual-circuit services because the lifetimes of connections are typically larger than the routing update interval.

In the last part of the dissertation, we formulate a quasi-static datagram model to evaluate a new approach to providing different type-of-service (TOS) classes of best-effort service. Instead of the traditional first-in-first-out (FIFO) link scheduling, our approach uses a class-based round-robin discipline and exploits this structure when calculating link costs.

Our quasi-static assumption makes the model analytically tractable while capturing the important dynamics and interactions between routing and scheduling. We apply the control-theoretic Liapunov function method to obtain the set of system states that lead to the optimal state, and demonstrate that this set is larger with our approach than with the traditional FIFO-based approach.

1.2 Organization of the Dissertation

Chapter 2 presents the Z-iteration method for a general dynamic MCMR model. In Chapter 3, we apply the Z-iteration to three specific systems with time-varying inputs and dynamic control, namely, an integrated network, a parallel database server, and a distributed batch system. Validations against discrete-event simulations are given in Chapter 4. Chapter 5 applies the Z-iteration to a detailed network model, and investigates three routing schemes on the NSFNET backbone topology. Chapter 6 describes approaches to providing classes of best-effort service in a datagram TOS network, and presents a quasi-static model. Analysis of this quasi-static model to compare the various approaches is given in Chapter 7. Chapter 8 concludes and identifies future research directions.

1.3 Publications

Most of the work presented in Chapters 2, 3 and 4 appears in [77]. The work presented in Chapter 5 appears in [74]. The work presented in Chapters 6 and 7 appears in [75, 76].

Chapter 2

$\mathcal{Z}$-Iteration: An Evaluation Method for Dynamic Models

2.1 Introduction

We consider a general multiple-class multiple-resource (MCMR) system. We have a set $\mathcal{R}$ of resources and a set $\mathcal{C}$ of customer classes. The nature of a resource depends on the system being modeled; for example, it may be computer memory, floor space, transmission capacity, etc. Each resource r has an attribute, denoted by $r.max$, which is a constant that indicates the maximum number of units in terms of which r is quantified.

Each class in $\mathcal{C}$ represents a class of customers that requires a particular set of resources. Depending on the system being modeled, customers can be user programs, manufactured products, network connections (calls), etc. Specifically, each class-c customer requires some subset $\mathcal{R}_c$ of resources, $\mathcal{R}_c \subseteq \mathcal{R}$. Furthermore, the class-c customer requires some number of units, denoted by $c.r.req$, of each resource $r \in \mathcal{R}_c$ (e.g. bandwidth, storage space, etc.). For example, a network connection would require some transmission and buffer capacity on each of the links of the path connecting its source to its destination.

Let $\lambda_c(t)$ denote the instantaneous arrival rate of class-c requests, and $1/\mu_c^r(t)$ denote the instantaneous service (or processing) time of a class-c request at r. Thus we are interested not only in the steady-state behavior of the MCMR system, but also in its transient or non-stationary behavior. Transient conditions arise when the statistics of the customer arrival processes or the service rates of the resources vary with time, due to externally time-varying factors or dynamic control decisions based on current or delayed system state information.

An arriving class-c customer is blocked at a resource $r \in \mathcal{R}_c$ iff $c.r.req$ exceeds the amount of the resource that is currently available (additional constraints can be incorporated too). An arriving class-c customer is blocked iff it is blocked at any $r \in \mathcal{R}_c$. A blocked customer is lost or retried later. Among the main performance measures of interest are the instantaneous blocking probabilities (or equivalently the throughputs) of the different classes, instantaneous average number of customers at resources, etc.

The generality of our model allows us to consider a variety of systems, including those with delayed feedback between changes in system state information and changes in control decisions. Examples of such systems include database locking systems, inventory systems, distributed batch systems, manufacturing systems, and communication networks. Because the class of a customer can be assigned when the customer arrives, it is straightforward to model state-dependent control policies such as assigning jobs to processors with the least workload.

MCMR systems have often been analyzed under steady-state conditions (e.g. [53, 57, 70, 29, 92, 21, 80, 43]). In this chapter, we formulate a dynamic flow model [35] to account for transient conditions as well. We solve our model by an iteration that differs from iterations commonly used in steady-state analysis, which only solve for steady-state measures and ignore the effect of delayed feedback.

Our solution method

The generality and time-dependency of our model seem to preclude analytical closed-form solutions. Our solution method, referred to as *Z-iteration* is, however, numerical. This has significant computational advantages over the (straightforward) discrete-event simulation approach, which requires the averaging of a large number of independent simulation runs to obtain meaningful performance estimates.

The instantaneous blocking probability of a class c, denoted by $B_c(t)$, is defined as the instantaneous probability that a class-c request is blocked at any of the required $\mathcal{R}_c$ resources. For this, we decompose our MCMR system into a set of multiple-class single-resource (MCSR) systems by invoking the resource independence assumption. Denoting by $B_c^r(t)$ the instantaneous blocking probability of class c at resource r, we have $B_c(t) = 1 - \prod_{r \in \mathcal{R}_c}(1 - B_c^r(t))$.

The Z-iteration computes $B_c^r(t)$ together with $N_c^r(t)$, the instantaneous average number of class-c requests waiting or in service at resource r, and $U_c^r(t)$, the instantaneous average number of class-c requests in service at resource r.

Let the index c' range over the set of classes requiring resource r. The Z-iteration depends upon the availability of two steady-state results about each MCSR system $r \in \mathcal{R}$ assuming that the $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ are constants: (1) an expression for the steady-state blocking probability B_c^r in terms of the steady-state actual offered loads $\lambda_{c'}/\mu_{c'}^r$; and (2) an expression for the steady-state utilization U_c^r in terms of the steady-state average numbers of customers $N_{c'}^r$, from which we readily obtain an expression for λ_c/μ_c^r in terms of the $N_{c'}^r$ and B_c^r.

These two steady-state results are available for a variety of MCSR systems, including self-service systems where the customer is also the server, and single- or multiple-server queueing systems [63, 35]. We point out that the expressions do not have to be closed form and can be implicit.

We make use of the concept of *instantaneous fictitious offered load*, originally introduced in [37], to obtain instantaneous versions of the above expressions. Specifically, we replace B_c^r by $B_c^r(t)$, the $N_{c'}^r$ by $N_{c'}^r(t)$, and the $\lambda_{c'}/\mu_{c'}^r$ by instantaneous fictitious offered loads $z_{c'}^r(t)$. (Note that λ_c/μ_c^r is not replaced by $\lambda_c(t)/\mu_c^r(t)$.)

This yields for every $r \in \mathcal{R}$ two "instantaneous" expressions, one for $B_c^r(t)$ in terms of the $z_{c'}^r(t)$, and one for $z_c^r(t)$ in terms of the $N_{c'}^r(t)$ and $B_c^r(t)$. A third instantaneous expression is obtained from standard flow balance, defining $N_c^r(t+\delta)$ in terms of the $N_{c'}^r(t)$, $\lambda_c(t)$, $\mu_c^r(t)$, and $B_c^{r'}(t)$ for $r' \in \mathcal{R}_c$, where δ is the time step for computing the instantaneous measures. With these three instantaneous expressions we compute the $B_{c'}^r(t)$, $z_{c'}^r(t)$, and $N_{c'}^r(t+\delta)$ in terms of the $N_{c'}^r(t)$, $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ for $t = 0, \delta, 2\delta, \cdots$. Specifically, given the $N_{c'}^r(t)$, we iterate over the first two expressions until the $B_{c'}^r(t)$ and $z_{c'}^r(t)$ converge. Then we use the third expression to compute the $N_{c'}^r(t+\delta)$.

Section 2.2 presents the Z-iteration method for the general MCMR model. Section 2.3 discusses its convergence. Section 2.4 discusses related work.

2.2 The Method

Figure 2.1 outlines our solution method to the general MCMR model introduced in Section 2.1. Recall that the following measures have been introduced:

- $B_c^r(t)$, instantaneous blocking probability of class c at resource $r \in \mathcal{R}_c$.
- $N_c^r(t)$, instantaneous average number of class-c requests waiting or in service at resource r.
- $U_c^r(t)$, instantaneous utilization of resource r by class-c requests (average number of class-c requests in service at resource r).
- $z_c^r(t)$, instantaneous fictitious offered load of class-c requests at resource r.

Let $\mathcal{C}^r$ denote the set of classes requesting units of resource r. In the outermost iteration, we obtain $\{N_c^r(t+\delta),\ B_c^r(t) :\ r \in \mathcal{R},\ c \in \mathcal{C}^r\}$ for $t = 0,\ \delta,\ 2\delta,\ \cdots$. The computation for each time t consists of two parts. The first part (steps 3-9) computes, for every $r \in \mathcal{R}$, $\{B_c^r(t)\text{: } c \in \mathcal{C}^r\}$ in terms of $\{N_c^r(t)\text{: } c \in \mathcal{C}^r\}$. The second part (step 10) computes, for every $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$, $N_c^r(t+\delta)$ in terms of $\{N_{c'}^r(t)\text{: } c' \in \mathcal{C}^r\}$, $\lambda_c(t)$, $\mu_c^r(t)$, and $\{B_c^{r'}(t)\text{: } r' \in \mathcal{R}_c\}$. The first part involves an iterative procedure (steps 5-9) on instantaneous versions of two steady-state formulas (steps 7 and 8). We describe these in detail below. The idea here is that *the instantaneous relationship between* $\{B_c^r(t)\text{: } c \in \mathcal{C}^r\}$ *and* $\{N_c^r(t)\text{: } c \in \mathcal{C}^r\}$ *is very well approximated by their relationship at steady-state.*

We define a *feasible state* of resource r by the number of requests from each class $c \in \mathcal{C}^r$ that r can simultaneously support, i.e. for which the total number of units requested does not exceed $r.max$. Let $\mathcal{F}^r$ denote the set of all feasible states of r.

```
1. Initialize {N_c^r(0) : r ∈ R, c ∈ C^r}            /* 0 for initially empty system */
2. For t = 0, δ, 2δ, ···
     begin
3.     For every r ∈ R      /* Obtain {B_c^r(t) : c ∈ C^r} in terms of {N_c^r(t) : c ∈ C^r} */
         begin
4.         Initialize {ẑ_c^r(t) : r ∈ R, c ∈ C^r}          /* arbitrary value if t = 0 */
                                                           /* ẑ_c^r(t − δ) if t > 0 */
5.         repeat
6.           z_c^r(t) ← ẑ_c^r(t), for every c ∈ C^r
7.           Obtain {B_c^r(t) : c ∈ C^r} in terms of {z_c^r(t) : c ∈ C^r}
                 using an instantaneous version of a steady-state formula (see (2.2))
8.           Obtain {ẑ_c^r(t) : c ∈ C^r} in terms of {B_c^r(t), N_c^r(t) : c ∈ C^r}
                 using an instantaneous version of a steady-state formula (see (2.4))
9.         until | ẑ_c^r(t) − z_c^r(t) | < ε, for every c ∈ C^r
         end
10.    For every r ∈ R and c ∈ C^r,
         obtain N_c^r(t + δ) in terms of {N_{c'}^r(t) : c' ∈ C^r}, λ_c(t), μ_c^r(t), and {B_c^{r'}(t) : r' ∈ R_c}
              using a difference equation relating arrivals and departures (see (2.5))
     end
```

Figure 2.1: Evaluation method.

Details of step 7

The first steady-state formula expresses the steady-state blocking probability B_c^r of class c at resource r in terms of the steady-state actual offered loads $\{\lambda_{c'}/\mu_{c'}^r : c' \in \mathcal{C}^r\}$. That is, assuming the $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ are constants for all t, the steady-state transition rate between two states belonging to $\mathcal{F}^r$ is given by some function of $\lambda_{c'}$ and $\mu_{c'}^r$. A class-c request is blocked in a state of $\mathcal{F}^r$ if its admittance would lead to a state outside $\mathcal{F}^r$. Refer to such states of $\mathcal{F}^r$ as class-c blocking states. We solve analytically for the probability of being in a class-c blocking state, yielding a formula $\mathcal{S}_c^r$ in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$:

$$B_c^r = \mathcal{S}_c^r(\{\frac{\lambda_{c'}}{\mu_{c'}^r} : c' \in \mathcal{C}^r\}) \quad \text{for} \quad c \in \mathcal{C}^r \tag{2.1}$$

To illustrate, consider an $M/G/m/m$ resource used by one class of customers arriving according to a Poisson process of rate λ_c. Let each admitted customer be served by one of the m servers for an average duration of $1/\mu_c^r$. Then $\mathcal{S}_c^r$ is the Erlang-B formula, i.e. $\mathcal{S}_c^r = E(\frac{\lambda_c}{\mu_c^r}, m) = \frac{(\frac{\lambda_c}{\mu_c^r})^m/m!}{\sum_{j=0}^{m}(\frac{\lambda_c}{\mu_c^r})^j/j!}$ [63].

The instantaneous version of (2.1) is obtained by replacing B_c^r by $B_c^r(t)$ and $\frac{\lambda_{c'}}{\mu_{c'}^r}$ by $z_{c'}^r(t)$, yielding

$$B_c^r(t) = \mathcal{S}_c^r(\{z_{c'}^r(t) : \ c' \in \mathcal{C}^r\}) \quad \text{for} \ \ c \in \mathcal{C}^r \tag{2.2}$$

Details of step 8

The second steady-state formula, which we refer to as $\mathcal{T}_c^r$, expresses U_c^r, the steady-state utilization of resource r by class-c customers, in terms of $\{N_{c'}^r : \ c' \in \mathcal{C}^r\}$, the steady-state average numbers of customers at resource r:

$$U_c^r = \mathcal{T}_c^r(\{N_{c'}^r : \ c' \in \mathcal{C}^r\})$$

From this and $\mu_c^r \, U_c^r = \lambda_c \, [1 - B_c^r]$, obtained by equating the departure rate to the admission rate, we have

$$\frac{\lambda_c}{\mu_c^r} = \frac{\mathcal{T}_c^r(\{N_{c'}^r : \ c' \in \mathcal{C}^r\})}{[1 - B_c^r]} \quad \text{for} \ \ c \in \mathcal{C}^r \tag{2.3}$$

$\mathcal{T}_c^r$ is a function that reflects the load and service discipline of r. The exact form of $\mathcal{T}_c^r$ is application dependent. One approximation to obtain $\mathcal{T}_c^r$ is to assume no blocking and then use steady-state queueing formulas expressing N_c^r in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$. Inverting these formulas, we obtain $\frac{\lambda_c}{\mu_c^r}$ in terms of the $N_{c'}^r$. Since we are assuming no blocking, from equation (2.3), we have $\mathcal{T}_c^r = \frac{\lambda_c}{\mu_c^r}$. Thus, we get $\mathcal{T}_c^r$ in terms of the $N_{c'}^r$. For example, consider a self-service facility where the customer is also the server, as in an $M/G/m/m$ queueing system. Assuming no blocking, we know that for the $M/G/\infty$ system $N_c^r = \frac{\lambda_c}{\mu_c^r}$ [63]. From this and $\mathcal{T}_c^r = \frac{\lambda_c}{\mu_c^r}$, which holds assuming no blocking, we have $\mathcal{T}_c^r = N_c^r$. Note that the approximation (due to the assumption of no blocking) is correct here for the self-service system where there is no waiting and by definition we directly have $\mathcal{T}_c^r = N_c^r$. (See Sections 2.3 and 3.2.)

The instantaneous version of (2.3) yields

$$z_c^r(t) = \frac{\mathcal{T}_c^r(\{N_{c'}^r(t) : \ c' \in \mathcal{C}^r\})}{[1 - B_c^r(t)]} \quad \text{for} \ \ c \in \mathcal{C}^r \tag{2.4}$$

Knowing $\{N_c^r(t)\colon \ c \in \mathcal{C}^r\}$ at some fixed t, we can solve equations (2.2) and (2.4) iteratively for $\{B_c^r(t)\colon \ c \in \mathcal{C}^r\}$. In particular, starting from an initial estimate $\{\hat{z}_c^r(t)\colon \ c \in \mathcal{C}^r\}$, we compute $\{B_c^r(t)\colon \ c \in \mathcal{C}^r\}$ from equations (2.2). Then, we use equations (2.4) to compute new values for $\{z_c^r(t)\colon c \in \mathcal{C}^r\}$. We repeat this process until the values of $\{z_c^r(t)\colon c \in \mathcal{C}^r\}$ stabilize as illustrated in steps 5-9 of Figure 2.1.

Details of step 10

At a fixed time t, once we obtain $\{B_c^r(t) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$, we obtain $\{N_c^r(t+\delta) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$, where δ is the discrete-time step, using the following difference equation:

$$N_c^r(t+\delta) = N_c^r(t) - \mu_c^r(t)\ U_c^r(t)\ \delta + \lambda_c(t)\ \delta \prod_{r' \in \mathcal{R}_c} [1 - B_c^{r'}(t)] \tag{2.5}$$

The second term in the right-hand side of equation (2.5) represents the average number of class-c requests which finish using (and depart from) resource r during $[t,\ t+\delta)$; the quantity $U_c^r(t)$ is computed from $\mathcal{T}_c^r(\{N_{c'}^r(t) :\ c' \in \mathcal{C}^r\})$. The third term represents the average number of new class-c requests that are admitted to resource r during $[t,\ t+\delta)$. Note that the product term $\prod$ reflects the assumption made in Section 2.1 that a new class-c request is admitted iff it is not blocked at any of the required $\mathcal{R}_c$ resources (this invokes the resource independence assumption).

Comments

Assuming that K iterations are needed for convergence of the iterative procedure in steps 5-9 of Figure 2.1, the computational complexity for each time step is $O(|\mathcal{R}|\ |\mathcal{C}^r|\ (\ (|B_c^r| + |z_c^r|)K + |N_c^r|\ ))$, where $|B_c^r|$ is the cost of evaluating $B_c^r(.)$ via (2.2), $|z_c^r|$ that of evaluating $z_c^r(.)$ via (2.4), and $|N_c^r|$ that of evaluating $N_c^r(.)$ via (2.5). The Z-iteration requires storage of $O(V\ |\mathcal{R}|\ |\mathcal{C}^r|)$, where V is the number of instantaneous measures. From Figure 2.1, we have $V = 5$ since we have 5 instantaneous measures defined, namely, $B_c^r(.)$, $z_c^r(.)$, $N_c^r(.)$, $\lambda_c(.)$ and $\mu_c^r(.)$.

We note that it might be required to make assumptions about the arrival or service distributions in order to obtain the $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ formulas.

Above we defined the feasible state of resource r by a multi-dimension vector representing the number of requests from each class $c \in \mathcal{C}^r$ that r can simultaneously support. In fact, we can define a feasible state differently as long as in this state, the total number of units requested does not exceed $r.max$. For example, we can define it by a single number representing the total number of units of r currently used by customers. Also, other criteria can used to further limit admission of requests.

The Z-iteration can also be used to directly solve for steady-state, if the $\lambda_c(t)$ and $\mu_c^r(t)$ are constants and a solution exists. We simply set $\frac{N_c^r(t+\delta) - N_c^r(t)}{\delta} = 0$ in equations (2.5) and use them in conjunction with equations (2.2) and (2.4) to iteratively solve for steady-state. There is no simple way to determine whether there exists a solution to such a nonlinear system. Even though the physical nature of the system usually suggests that a solution exists, the iteration may oscillate between different solutions, which can alert one to the instability of the system [21]. Obviously, such oscillations can also occur under transient conditions arising, for example, from dynamic control.

Observe that it is easy to realize parallel implementations of our method by mapping the computations for different resources onto different processors, and we would expect almost linear speedup.

2.3 Error and Convergence

The accuracy of our method depends on the approximation of the relationship between the $B_c^r(t)$ and the $N_c^r(t)$ by its steady-state counterpart, which is the fixed point of the iteration in steps 5-9 of Figure 2.1. Our experience indicates that our method yields accurate performance measures when compared to discrete-event simulation and that the iteration converges quickly (see Chapter 4).

Analyzing the errors and convergence of this iteration is hard in general. However, it can be shown in simple situations that the approximation is accurate when compared to the exact instantaneous solution, and that the iteration is a contractive mapping of [0, 1) into [0, 1) and hence it converges to a unique fixed point [61]. We show this for the $M/M/2/2$ system with constant arrival and service rates.

Accuracy of the approximation for the $M/M/2/2$ system

From [98], we have the following exact instantaneous solution. Consider the $M/M/2/2$ system initially empty with customer arrival rate $\lambda(t) = \lambda$ and service rate $\mu(t) = \mu$ for all t. The Laplace transform $P_n(s)$ of the state probability $P_n(t)$, where n denotes the number of customers in the system ($n = 0, 1, 2$), is given by

$$\begin{aligned} P_n(s) &= \int_0^\infty e^{-st} P_n(t)dt \\ &= \sum_{i=n}^{2} (-1)^{i-n} \begin{pmatrix} i \\ n \end{pmatrix} \beta_i(s) \end{aligned}$$

where

$$\beta_i(s) = \frac{\frac{1}{(s+i\mu)} \sum_{j=i}^{2} \begin{pmatrix} 2 \\ j \end{pmatrix} \frac{(s+i\mu)\cdots(s+j\mu)}{\lambda^{j+1-i}}}{\sum_{j=0}^{2} \begin{pmatrix} 2 \\ j \end{pmatrix} \frac{s(s+\mu)\cdots(s+j\mu)}{\lambda^{j+1}}}$$

From the above result, we can directly obtain the exact instantaneous expressions for the blocking probability $P_2(t)$, henceforth denoted by $B_{\text{exact}}(t)$, and for the average number of customers in the system, henceforth denoted by $N_{\text{exact}}(t)$. In particular, we use *Mathematica* [105] to obtain the inverse Laplace transforms of the following:

$$\begin{aligned} B_{\text{exact}}(s) &= P_2(s) = \beta_2(s) = \frac{\lambda^2}{\lambda^2 s + 2\lambda s(s+\mu) + s(s+\mu)(s+2\mu)} \\ N_{\text{exact}}(s) &= P_1(s) + 2P_2(s) = \beta_2(s)\left(\frac{s + 2\lambda + 2\mu}{\lambda}\right) \end{aligned}$$

The steady-state relationship between the blocking probability P_2, henceforth denoted by B_{ss}, and the average number of customers in the system, henceforth denoted by N_{ss}, is given by:

$$B_{\text{ss}} = \frac{\rho^2/2}{1 + \rho + \rho^2/2}, \text{ where } \rho = \lambda/\mu \qquad (2.6)$$

$$N_{\mathrm{ss}} = \frac{\rho + \rho^2}{1 + \rho + \rho^2/2} \tag{2.7}$$

Inverting equation (2.7), we get

$$\rho = \frac{N_{\mathrm{ss}} - 1 + \sqrt{1 + 2N_{\mathrm{ss}} - N_{\mathrm{ss}}^2}}{2 - N_{\mathrm{ss}}} \tag{2.8}$$

Substituting (2.8) in (2.6), we get B_{ss} in terms of N_{ss}.

To illustrate the accuracy of approximating the relationship between $B_{\mathrm{exact}}(t)$ and $N_{\mathrm{exact}}(t)$ by the relationship between B_{ss} and N_{ss}, we plot in Figure 2.2 $B_{\mathrm{exact}}(t)$ and $B_{\mathrm{ss}}\ |_{N_{\mathrm{ss}}=N_{\mathrm{exact}}(t)}$ for $\lambda = \mu = 1$ and $t \in [0, 6]$. Clearly, this shows that the approximation is quite good.

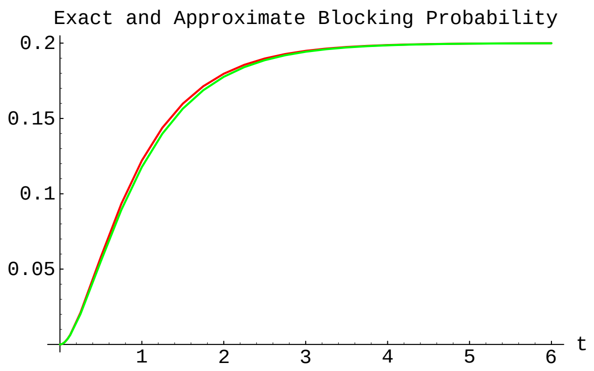

Figure 2.2: Accuracy of the approximation for the $M/M/2/2$ system.

Convergence of the iteration for the $M/M/2/2$ system

The iteration in steps 5-9 of Figure 2.1 implicitly defines the relationship between B_{ss} and N_{ss}. Actually for the $M/M/2/2$ system we do not need to iterate since we have (2.8), and consequently we have an explicit relationship between B_{ss} and N_{ss}. But for blocking systems in general, equation (2.7), from which (2.8) was obtained, is not invertible. So instead of equation (2.8) our method uses equation (2.3), where we obtain the steady-state utilization in terms of N_{ss} assuming a nonblocking system (this approximation turns out to be very good in general; see Chapter 4). Thus here we illustrate the convergence of the iteration defined by:

$$\begin{aligned} B_{\mathrm{ss}} &= \frac{\rho^2/2}{1 + \rho + \rho^2/2} \\ \rho &= \frac{N_{\mathrm{ss}}}{[1 - B_{\mathrm{ss}}]} \end{aligned}$$

These two formulas define an equation of the form $B_{\mathrm{ss}} = F(B_{\mathrm{ss}})$. Figure 2.3 shows a graphical example of the mapping F. It illustrates that F is a contractive mapping of $[0, 1)$ into $[0, 1)$ and hence it converges to a unique fixed point [61].

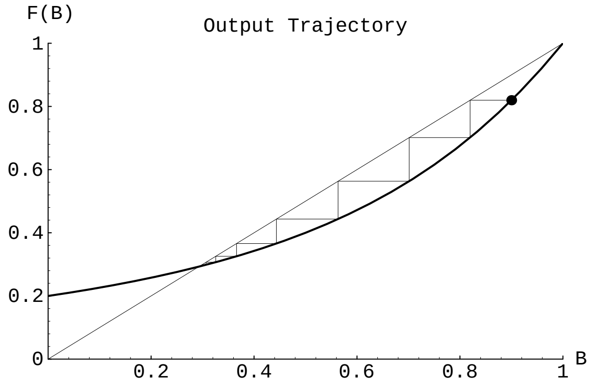

Figure 2.3: Convergence of the iteration for the $M/M/2/2$ system starting from $B_{\mathrm{ss}} = 0.9$ for $N_{\mathrm{ss}} = 1$.

2.4 Related Work

MCMR systems have often been analyzed under steady-state conditions (e.g. [53, 57, 70, 29, 92, 21, 80, 43]). In this chapter, we formulated a dynamic flow model [35] to account for transient conditions as well. We solved our model by an iteration that differs from iterations commonly used in steady-state analysis, which only solve for steady-state measures and ignore the effect of delayed feedback.

Our model yields the time-varying behavior of a general MCMR system. We use the well-known decomposition technique [62, 57] to approximate the system as a collection of MCSR systems. For each MCSR system, we describe the evolution of the instantaneous average number of customers of each class by relating its instantaneous admission rate to its instantaneous departure rate. The computation of these instantaneous rates uses a basic concept, that of approximating instantaneous relationships by their steady-state counterparts.

To obtain the instantaneous admission rates, we adapt steady-state queueing formulas to yield the instantaneous blocking probability of each class in terms of the instantaneous average numbers of customers waiting and in service. This uses the technique of fictitious offered load. The technique was originally introduced in [37], where it was used to obtain steady-state blocking probability and carried load for a specific call routing and network topology.

Reference [37] considered a network of source nodes, destination nodes, and intermediate nodes, with a link from every source node to every intermediate node, and a link from every intermediate node to every destination node. Each link can carry a fixed total number of calls. The call arrival process from a source to a destination is Poisson with fixed rate. The call routing is not dynamic; a fixed fraction of the call arrivals is routed through every intermediate node. In addition, overflow traffic (due to blocking links) is routed through alternate available routes. Each call, once admitted, has an exponential holding time of fixed mean that is the same for all calls. The blocking probability of a link is given by the Erlang-B formula expressed in terms of fictitious combined offered load. The system is solved for steady-state average number of calls on each link by equating the call departure rate to the call admission rate.

Our model extends this fictitious offered load technique to general multi-class systems, where, for example, each class has different resource and service needs, and resources have different scheduling disciplines. Also, our model can be applied to describe general dynamic routing schemes with the arrival rate of a class changing as a function of the instantaneous system state.

To obtain the instantaneous departure rates, we again adapt steady-state queueing formulas to yield the instantaneous utilization of each class in terms of the instantaneous average numbers of customers waiting and in service. The same technique was used in [100], where feedforward queueing networks were considered. Each service center is an $M/M/1$ infinite FCFS queue with the same average service time for all classes. The routing of each class is a time-dependent Bernoulli process. Compared to our model, this does not model blocking resources, or service centers with complicated structure (e.g. service centers consisting of multiple resources with different scheduling disciplines serving customers with different needs). Though we do not consider here sequential resource needs by one customer (a customer requests all needed resources simultaneously), our model is easily extended to capture this situation.

Our dynamic flow model is quite general, and can be used to study both transient and steady-state performances of various MCMR blocking and non-blocking systems. Our method has advantages over other methods that might be used to analyze transient behaviors. One such method is that of time-dependent queueing models, which involve probability distributions for all events. However, such models are extremely difficult to solve analytically [101], and computationally expensive to solve numerically [100]; A second method is that of diffusion models, which utilize averages and variances [19, 84]. Such models involve partial differential equations and are usually intractable. A third method is that of fluid models, which utilize average quantities only [15]. Such models involve ordinary differential equations and are usually tractable. However, dynamic flow models appear more accurate since they include detailed probabilistic descriptions manifested in our model in the computation of both the instantaneous blocking probabilities and the instantaneous utilizations.

Chapter 3

Simple Applications of the Z-Iteration

In Chapter 2, we described the Z-iteration for a general MCMR model. As we pointed out, the exact form of $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ and the values of $\mu_c^r(.)$ and $\lambda_c(.)$ depend on the particular application. In this chapter, we consider different applications, and show how they fit into the general model and solution procedure. We consider an integrated network, a parallel database server, and a distributed batch system. The first and third systems are modeled as systems with self-service resources, for which validations against discrete-event simulations are given in Section 4.1. The second system is modeled as a system with single-server resources, for which validations are given in Section 4.2.

3.1 Integrated Network Example

Consider an integrated network carrying various classes of connections. (See Figure 3.1.) A class represents connections with the same traffic and QoS parameters and routed on the same path from a source node to a destination node. The connections of a class c arrive according to a Poisson process of rate $\lambda_c(t)$. Each connection, once it is successfully setup, has a lifetime of average duration $\frac{1}{\mu_c(t)}$.

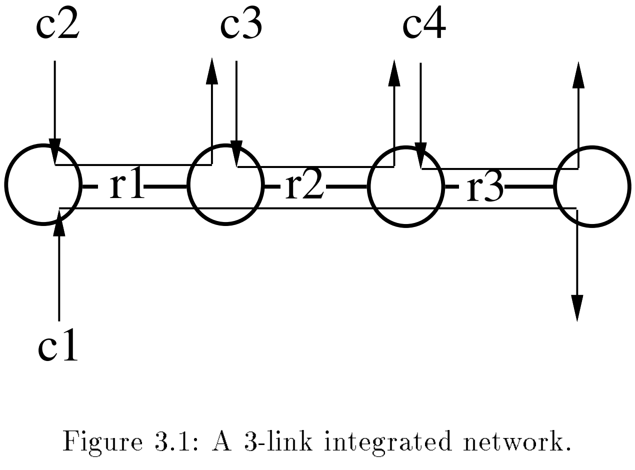

Figure 3.1: A 3-link integrated network.

Resources in a network include link bandwidths, buffer spaces, etc. For this example, we assume link bandwidths are the main resources; thus $\mathcal{R}$ consists of link ids (where each id denotes the bandwidth component of the link). We assume a connection requires the reservation of a certain amount of bandwidth on each link along its route that are enough to satisfy its QoS. This reservation amount can be thought of as either the peak transmission rate of the connection or its "effective bandwidth" [44] varying between its peak and average transmission rates.

The set $\mathcal{R}_c$ of a class-c connection would thus contain the links along the route of class c. An arriving class-c connection that finds insufficient bandwidth on any $r \in \mathcal{R}_c$ is blocked and lost. Otherwise, the connection is admitted and bandwidths are allocated to it on each $r \in \mathcal{R}_c$ for an average duration of $\frac{1}{\mu_c^r(t)} = \frac{1}{\mu_c(t)}$. Note that this is a self-service system.

Thus, $r.max$ is the total link bandwidth of r, and $c.r.req$ is the amount of link bandwidth that must be allocated (reserved) for a class-c connection on $r \in \mathcal{R}_c$. Let's assume that the $c.r.req$ and $r.max$ are integers. Let the state of r indicate the amount of bandwidth allocated. Thus, $\mathcal{F}^r = \{0, 1, \cdots, r.max\}$. Let $Q^r(j)$ denote the steady-state probability of r being in state j. Then the $Q^r(.)$ satisfy the following recurrence relation [92]:

$$j\,Q^r(j) = \sum_{c' \in \mathcal{C}^r} \frac{\lambda_{c'}}{\mu^r_{c'}}\, c'.r.req\; Q^r(j - c'.r.req) \qquad j = 1, \ldots, r.max$$

where $\sum_{j=0}^{r.max} Q^r(j) = 1$.

The steady-state blocking probability for class-c connections at r, B_c^r, is given by

$$B_c^r = \sum_{j=r.max-c.r.req+1}^{r.max} Q^r(j)$$

This steady-state solution, which defines $\mathcal{S}_c^r(.)$ for this system, is valid for Poisson arrivals and general service times. It can be used in equations (2.2) after replacing the $\frac{\lambda_{c'}}{\mu^r_{c'}}$ by fictitious offered loads $z^r_{c'}(t)$.

Regarding the function $\mathcal{T}_c^r(.)$ used in equations (2.4), since r is self-service, we have

$$\mathcal{T}_c^r(.) = N_c^r(t)$$

In Chapter 5, we consider a detailed integrated network model and illustrate how the Z-iteration can capture the effects of various dynamic control schemes.

Systems with self-service resources are validated (against discrete-event simulations) in Section 4.1. There we consider systems equivalent to single-link network, and multi-link network. The multi-link network is used by several multi-hop connections representing main traffic, and several one-hop connections representing cross-traffic.

3.2 Parallel Database Server Example

Consider a system of multiple disks on which data is partitioned according to some scheme, e.g. round-robin, range partitioning, etc. [27]. Each disk has a finite first-come-first-served (FCFS) queue where queries of different classes wait to be served. A query requests data retrieval from one or more disks in parallel. (See Figure 3.2.) This parallelism typically leads to reduction in data access time [27, 54]. The collection of disks needed by a query is defined by the query's class. We assume an arriving query requires one unit of space in the queue of each disk it needs to access.

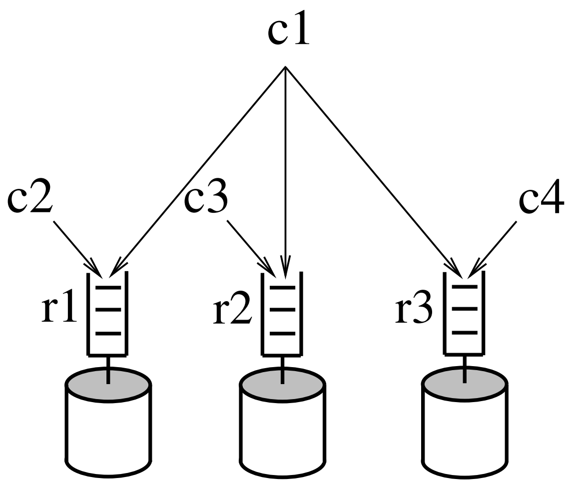

Figure 3.2: A 3-disk parallel database server.

Thus the resource set $\mathcal{R}_c$ of a class-c query contains the queues of disks that are needed by class c, and this is a function of the data partitioning scheme. $r.max$ is the total number of requests that r can accommodate, and $c.r.req = 1$ for $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$. An arriving class-c query that finds no space in any $r \in \mathcal{R}_c$ is blocked and lost.

Assume class-c queries arrive according to a Poisson process of rate $\lambda_c(t)$. Also, assume that the service time of any query in r is exponentially distributed with mean $\frac{1}{\mu^r}$; thus $\frac{1}{\mu_c^r(t)} = \frac{1}{\mu^r}$ for all $c \in \mathcal{C}^r$.

Let the state of r denote the total number of queries waiting or in service in r. Thus, $\mathcal{F}^r = \{0, 1, \ldots, r.max\}$. The steady-state blocking probability for class-c queries at r is the steady-state probability of r being in state $r.max$. This steady-state solution is well-known for the $M/M/1/r.max$ queueing system, in particular, for $c \in \mathcal{C}^r$:

$$B_c^r = \frac{\left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^{r.max}}{\sum_{j=0}^{r.max} \left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^j} \quad [63]$$

This steady-state solution can be used in equations (2.2) after replacing $\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}$ by $\sum_{c' \in \mathcal{C}^r} z_{c'}^r(t)$.

We employ the technique introduced in Section 2.2 to derive the function $\mathcal{T}_c^r(.)$ used in equations (2.4). Assuming steady-state and no blocking, we can treat the $M/M/1/r.max$ system of r as an $M/M/1/\infty$ system. At steady-state, we know that [63]

$$N_c^r = \frac{\lambda_c}{\mu^r - \sum_{c' \in \mathcal{C}^r} \lambda_{c'}} \tag{3.1}$$

From this and $\mathcal{T}_c^r(.) = \frac{\lambda_c}{\mu^r}$, which holds assuming no blocking, we have[1]

$$\mathcal{T}_c^r(.) = \frac{N_c^r}{1 + \sum_{c' \in \mathcal{C}^r} N_{c'}^r}$$

Therefore, in the transient regime, we have

$$\mathcal{T}_c^r(.) = \frac{N_c^r(t)}{1 + \sum_{c' \in \mathcal{C}^r} N_{c'}^r(t)}$$

The above model can be used to study various data partitioning schemes for high-performance indexing [27]. Systems with single-server resources are validated (against discrete-event simulations) in Section 4.2.

3.3 Distributed Batch System Example

Consider a distributed batch system such as Condor [69]. Batch jobs (user programs) are submitted to a central manager (CM). Assume batch jobs of type i arrive to the CM according to a Poisson process of rate λ_i. The CM uses its information about the load on the various workstations to choose for the arriving batch job a potential workstation for its execution. The class of the batch job is defined by the workstation it is routed to by the CM and the job type.

Each batch job would typically require resources such as memory, disk space, and CPU processing power to execute on a workstation. For this example, we assume all required resources other than the CPU are always available. The set $\mathcal{R}_c$ of a class-c batch job would thus contain the CPU of the workstation to which the job is routed.

We assume only one job can be running on each workstation at a time. Thus, if the owner of the workstation executes a job of his/her own, then the batch job currently executing on his/her workstation, if any, is suspended and its execution resumed later when the owner job finishes execution. An arriving class-c batch job that finds another batch job running or suspended on $r \in \mathcal{R}_c$ is blocked and returned to the CM. Otherwise, it is admitted for processing with mean

[1] From (3.1), we have (i) $N_c^r = \frac{\lambda_c/\mu^r}{1-\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}$, and thus (ii) $\sum_{c' \in \mathcal{C}^r} N_{c'}^r = \frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}{1-\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}$. Rearranging the last equation, we have (iii) $\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r = \frac{\sum_{c' \in \mathcal{C}^r} N_{c'}^r}{1+\sum_{c' \in \mathcal{C}^r} N_{c'}^r}$. Substituting (iii) in (i), we get an expression for $\frac{\lambda_c}{\mu^r}$, which together with $\mathcal{T}_c^r(.) = \frac{\lambda_c}{\mu^r}$ yields the desired result.

processing time of $1/\mu_c^r(t)$. This processing time includes the time during which the batch job is suspended due to owner processes [68]. Note that in this application, we do not assume that blocked jobs are lost, rather they are returned to the CM for retry.

The instantaneous arrival rate of class-c batch jobs of type i, $\lambda_c(t)$, is a function of λ_i, the load balancing algorithm used by the CM, and the rate of retrials of type i batch jobs. Assume the load balancing algorithm regularly assigns to the candidate workstations probabilities according to their measured loads. Arriving batch jobs are routed independently according to these probabilities. Let $\alpha_c(t)$ denote the load-dependent probability that the type i batch job belongs to class c, i.e. is routed to $r \in \mathcal{R}_c$. Then,

$$\lambda_c(t) = [\lambda_i \; + \sum_{\substack{\text{classes } c' \text{ of type } i \\ r' \in \mathcal{R}_{c'}}} \lambda_{c'}(t-\delta)\; B_{c'}^{r'}(t-\delta)]\; \alpha_c(t) \tag{3.2}$$

The $\sum$ term in equation (3.2) represents the total rate of retrials of type i batch jobs, which is a function of their blocking probabilities. In this model, $r.max$ is the maximum number of batch jobs that r handles. $r.max = 1$ and $c.r.req = 1$ for $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$. Let the state of r denote the total number of batch jobs running or suspended on r. Then, $\mathcal{F}^r = \{0, 1\}$. This system is similar to the self-service integrated network discussed in Section 3.1, and hence we can use the $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ formulas presented there.

Indeed, we are assuming here the arrival processes are Poisson. This is not, in general, true since the composite traffic contains blocked batch jobs returned immediately at the next time step to the system for retry. This assumption is less restrictive if blocked batch jobs are returned to the system after waiting an independent random period [80, 39]. This waiting effect can be easily incorporated into the above model. This model can be used to study the interactions between owner jobs and batch jobs, and examine various load balancing schemes through the $1/\mu_c^r(t)$ and $\alpha_c(t)$.

Chapter 4

Validation of the Z-Iteration

In this chapter, we present numerical results to validate the Z-iteration. Section 4.1 contains validations against discrete-event simulations for systems with self-service resources. Section 4.2 contains validations for systems with single-server resources.

4.1 Validation of Systems with Self-Service Resources

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for systems with self-service resources. In our method, we obtain instantaneous performance measures through equations (2.2), (2.4), and (2.5), substituting with the appropriate application-dependent parameters and formulas. We take the discrete-time step δ to be 0.1.

The simulation model differs from our analytical model in that the actual events of arrival and processing of requests are simulated according to the specified probability distributions and system characteristics (i.e. service disciplines, admission policy, etc.). To obtain reliable performance estimates, a number of independent replications (i.e. simulation runs) must be carried out and averaged. In particular, let $X^{(i)}(t)$ denote a generic measure computed at time instant t in replication i, where t takes on the successive values $t_1, t_2, \cdots, t_k, \cdots$. Then, the mean value of this measure at particular time instant t_k is estimated as $\sum_{i=1}^{N} X^{(i)}(t_k)/N$, where N is the total number of replications. The larger N is, the more accurate the simulation estimates are [71]. In our simulations, the performance measures are computed for $t = 1, 2, 3, \cdots$.

The measures considered are precisely defined as they are introduced below. In all experiments, we start with empty systems. For the cases with $N = 50$, the observed mean of the simulation measures at various time instants typically show 95% confidence interval for a $\pm$ 10% range. For the cases with higher N, 95% confidence interval is obtained for a $\pm$ 3% range.

We first consider a MCSR system with a single resource r1 used by 10 customer classes whose parameters are shown in Table 4.1.

Class-c customers arrive at r1 according to a Poisson process of rate λ_c. The system is self-

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1}	30	0.125	5
c2	{r1}	15	0.5	1
c3	{r1}	50	0.2	2
c4	{r1}	10	0.1	2
c5	{r1}	40	0.125	1
c6	{r1}	25	0.5	0.5
c7	{r1}	30	1.0	0.5
c8	{r1}	10	0.0625	10
c9	{r1}	5	1.0	0.2
c10	{r1}	50	0.25	2

Table 4.1: Parameters of 10 classes using r1 with r1.max = 200.

service. In particular, an admitted class-c customer holds the acquired $c.\mathrm{r1}.req$ resource units for an exponential duration with mean $1/\mu_c$ before releasing them. This system is similar to a single-link integrated network modeled as in Section 3.1, and hence we use the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas presented there to obtain the performance measures by our method.

Figures 4.1, 4.2, and 4.3 show the time behavior of the total number of in-service customers, the fraction of resource units allocated, and the total throughput, respectively. The first measure denotes the total number of customers currently holding resource units, which is equal to $\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} N_{c'}^{\mathrm{r1}}(t)$ in our method. The second measure denotes the fraction of r1.max currently being held by customers, which is equal to $(\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} N_{c'}^{\mathrm{r1}}(t) \times c'.\mathrm{r1}.req)/\mathrm{r1}.max$ in our method. The third measure denotes the total current admission rate, which is equal to $\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} \lambda_{c'}(1 - B_{c'}^{\mathrm{r1}}(t))$ in our method. Generally, it is equal to $\sum_{c' \in \mathcal{C}} \lambda_{c'} \prod_{r' \in \mathcal{R}_{c'}} [1 - B_{c'}^{r'}(t)]$ for MCMR systems.

In our simulations, the first two measures displayed at time instant t ($t = 1, 2, 3, \cdots$) are simply the values of these measures as observed at t. The last measure, namely the total throughput, displayed at time instant t is defined to be the total number of customers admitted in the interval $[t-1,\ t)$.

Our method yields results very close to the exact values. In addition, we found our method much less time-consuming than simulation. This is especially because the latter requires the averaging of a large number of independent simulation runs. To give an idea of the computational savings, for this experiment, on a DECstation 5000/133, our method required around 6 seconds of execution time while the 50-run and 1000-run simulations required around 25 seconds and 8 minutes, respectively. The number of iterations required at each time step for convergence of the iterative procedure in steps 5-9 of Figure 2.1 is less than 6 iterations for $\epsilon = 10^{-5}$ and $\hat{z}_c^r(0) = \lambda_c/\mu_c^r$.

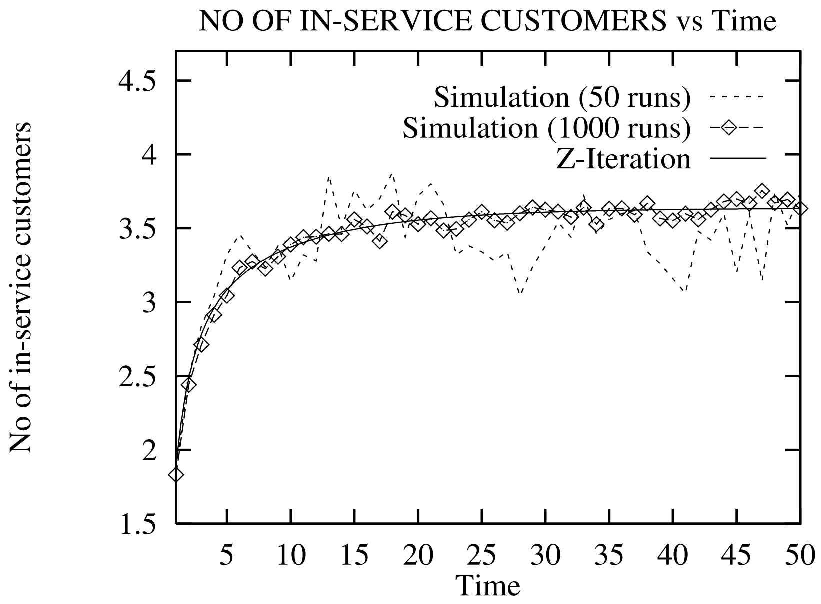

Figure 4.1: Total number of in-service customers versus time. MCSR self-service system.

We next validate our resource independence assumption manifested in equation (2.5) by the product term $\prod$. We consider a similar self-service system but with 3 resources and 20 customer classes. Out of the 20 classes, 10 classes require all 3 resources. A class-c customer requires the same number of units of each $r \in \mathcal{R}_c$. Table 4.2 shows the system parameters. Note that this system can be regarded as a multi-link integrated network modeled as in Section 3.1. See Figure 4.4. Here, classes 1 to 10 represent multi-hop connections modeling main traffic, while other classes represent one-hop connections modeling cross-traffic.

Figure 4.5 shows the instantaneous total throughput. Simulation results, denoted by Exp, are for Poisson arrivals and exponential holding times. Simulation results, denoted by Det, are for Poisson arrivals and deterministic holding times. The results show the accuracy of our method in both cases as they satisfy the assumptions required to obtain the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas used here. (Our experiments with deterministic arrivals show large errors as expected.)

Next, we consider a similar self-service system whose parameters are given in Table 4.3. Here, λ_{c1} varies with time. This mimics the effect of traffic control policies such as flow control and routing. We assume λ_{c1} alternates every 20 time units between zero and 0.125, starting with zero. Figures 4.6 and 4.7 show the instantaneous total throughput and blocking probability, respectively. Our method accurately reproduces the behavior obtained by simulation. We compute the instantaneous blocking probability $B(t)$ from the throughput $\gamma(t)$ using the relation $B(t) = 1 - \gamma(t)/\lambda(t)$, where $\lambda(t)$ is the instantaneous total arrival rate of requests. We do this rather than compute $B(t)$ directly from the simulations because doing that would require averaging over a very large number of replications, because $B(t)$ typically has a very low value and thus a high sample variance.

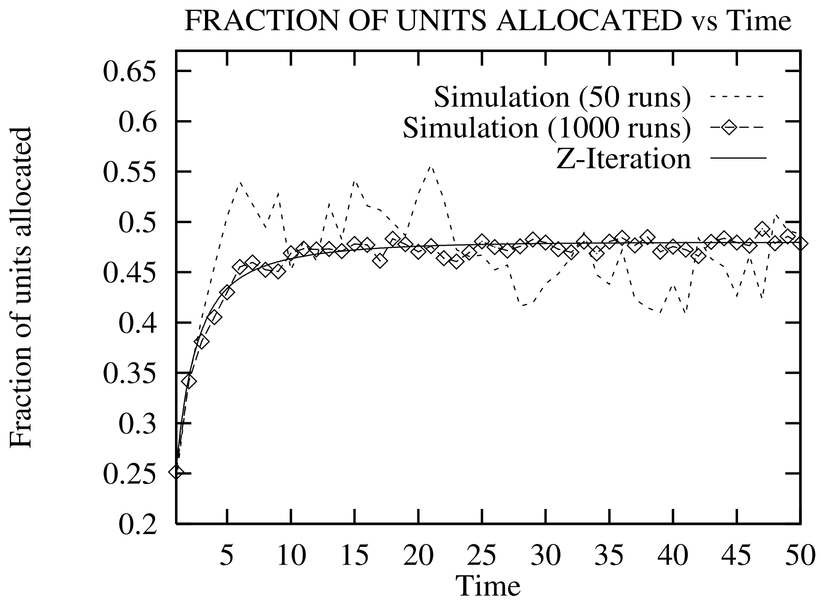

Figure 4.2: Fraction of resource units allocated versus time. MCSR self-service system.

4.2 Validation of Systems with Single-Server Resources

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for systems with single-server resources. The performance measures are computed as described in Section 4.1. Similar confidence intervals are also observed for the measures obtained by simulation.

We consider a MCMR system with 3 resources and 4 customer classes. Out of the 4 classes, class c1 requires all 3 resources. A class-c1 customer requires one unit of each resource. Table 4.4 shows the system parameters.

Class-c customers arrive according to a Poisson process of rate λ_c. Each resource consists of a single-server with a finite waiting room and a FCFS scheduling discipline. An admitted class-c customer occupies one unit of space, and requires an exponential service time with unit mean. This system is similar to the parallel database server discussed in Section 3.2, and hence we use the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas presented there to obtain the performance measures by our method. Figure 4.8 shows the instantaneous total throughput. The results obtained by our method agree with those obtained by simulation.

We next consider the same system but with λ_{c1} varying with time. We assume λ_{c1} alternates every 20 time units between zero and 0.2, starting with zero. Figures 4.9 and 4.10 show the instantaneous total throughput and blocking probability, respectively. Our method accurately reproduces the behavior obtained by simulation.

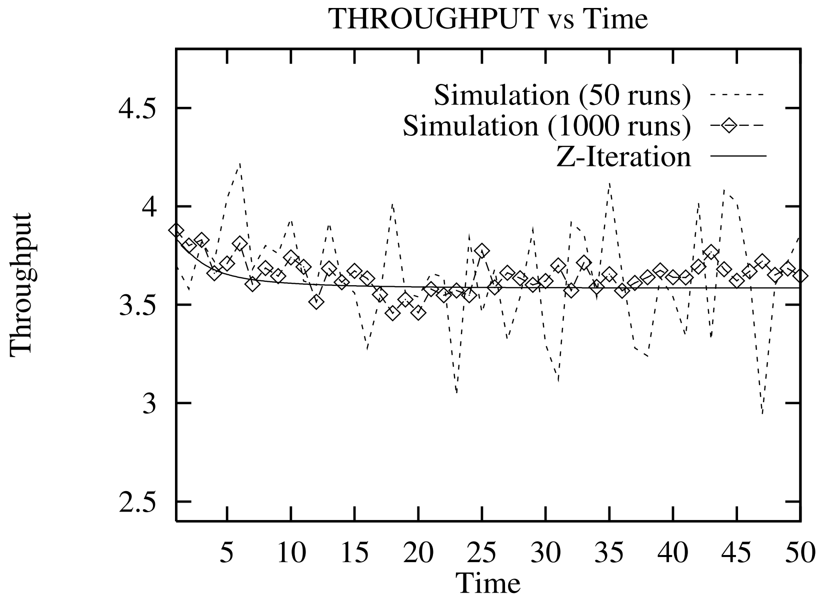

Figure 4.3: Total throughput versus time. MCSR self-service system.

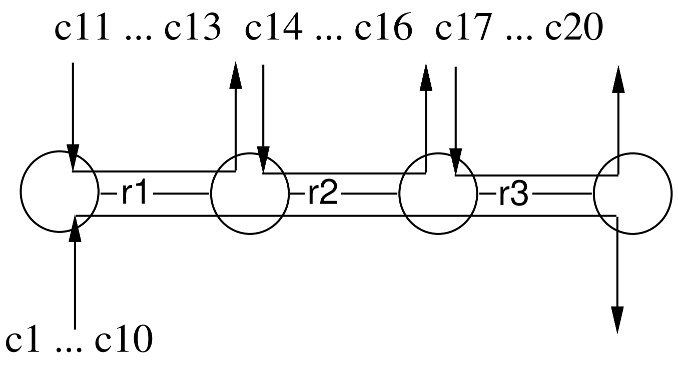

Figure 4.4: Multi-link network.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	30	0.125	5
c2	{r1, r2, r3}	15	0.5	1
c3	{r1, r2, r3}	50	0.2	2
c4	{r1, r2, r3}	10	0.1	2
c5	{r1, r2, r3}	40	0.125	1
c6	{r1, r2, r3}	25	0.5	0.5
c7	{r1, r2, r3}	30	1.0	0.5
c8	{r1, r2, r3}	10	0.0625	10
c9	{r1, r2, r3}	5	1.0	0.2
c10	{r1, r2, r3}	50	0.25	2
c11	{r1}	30	0.125	5
c12	{r1}	15	0.5	1
c13	{r1}	50	0.2	2
c14	{r2}	10	0.1	2
c15	{r2}	40	0.125	1
c16	{r2}	25	0.5	0.5
c17	{r3}	30	1.0	0.5
c18	{r3}	10	0.0625	10
c19	{r3}	5	1.0	0.2
c20	{r3}	50	0.25	2

Table 4.2: Parameters of 20 classes using 3 resources r1, r2, and r3 with r1.max = 150, r2.max = 200, and r3.max = 250.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	30	$0 \leftrightarrow 0.125$	5
c2	{r1}	30	0.125	5
c3	{r2}	10	0.1	2
c4	{r3}	50	0.25	2

Table 4.3: Parameters of 4 classes using 3 resources r1, r2, and r3 with r1.max = 50, r2.max = 100, and r3.max = 150.

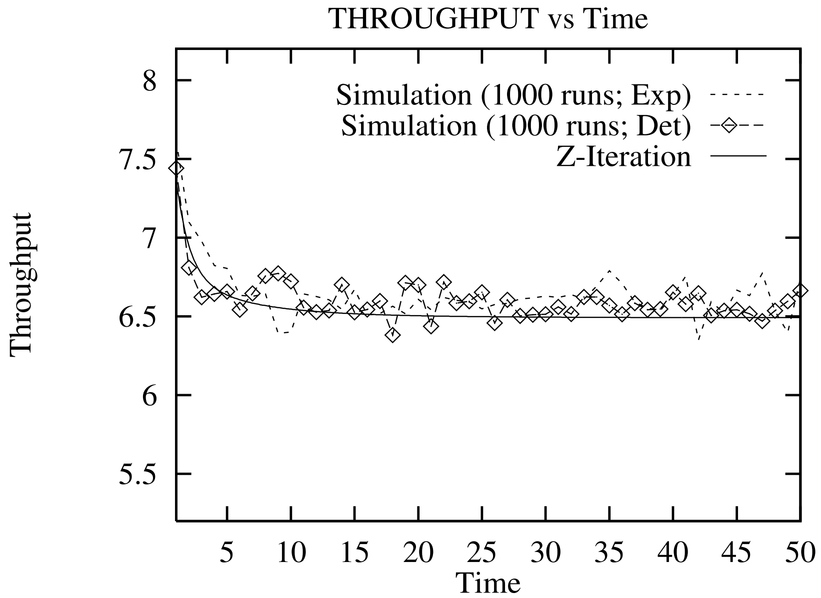

Figure 4.5: Total throughput versus time. MCMR system with self-service resources.

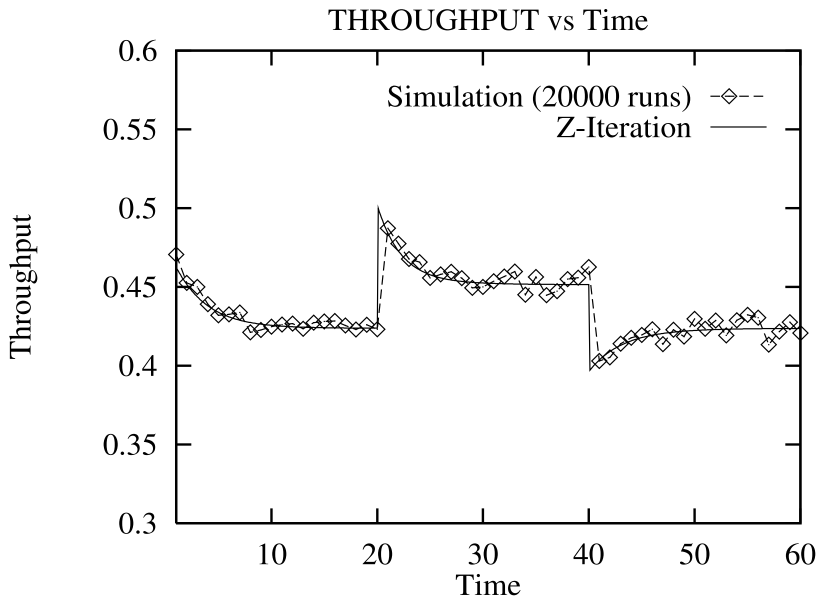

Figure 4.6: Total throughput versus time. MCMR system with self-service resources. Time-varying arrivals.

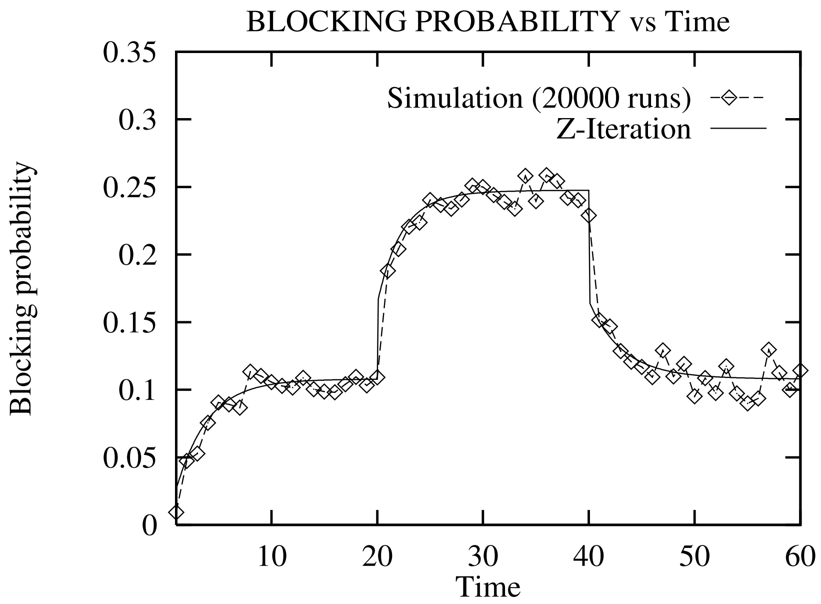

Figure 4.7: Blocking probability versus time. MCMR system with self-service resources. Time-varying arrivals.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	1	0.2	1
c2	{r1}	1	0.5	1
c3	{r2}	1	0.8	1
c4	{r3}	1	0.4	1

Table 4.4: Parameters of 4 classes using 3 resources with $r.max = 5$ each.

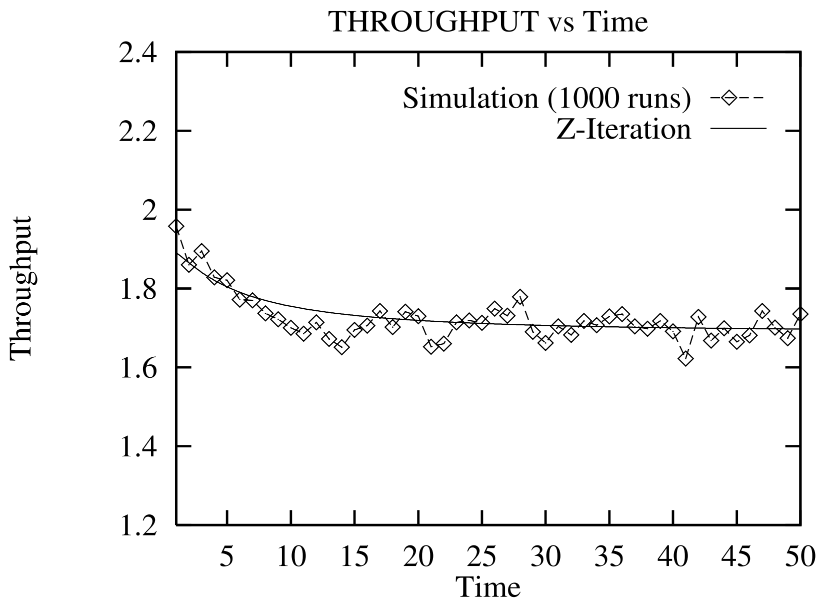

Figure 4.8: Total throughput versus time. MCMR system with single-server resources.

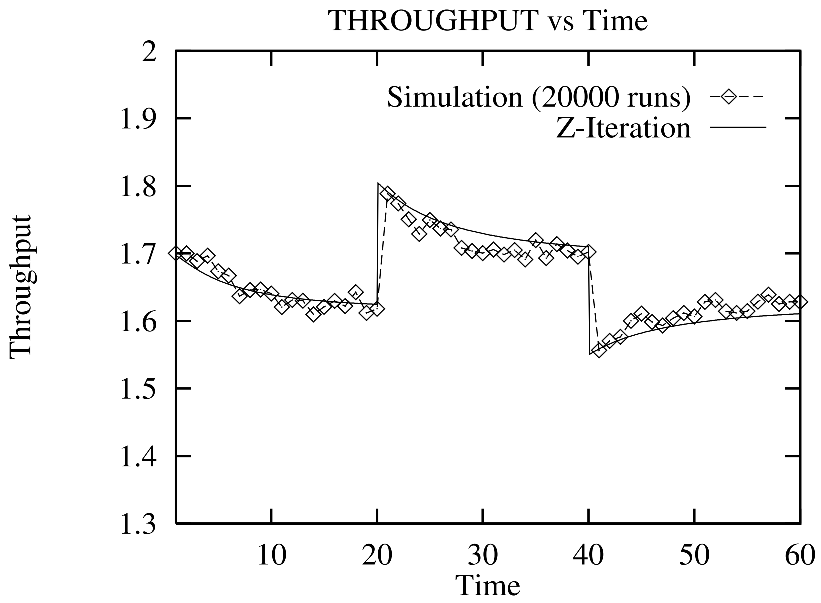

Figure 4.9: Total throughput versus time. MCMR system with single-server resources. Time-varying arrivals.

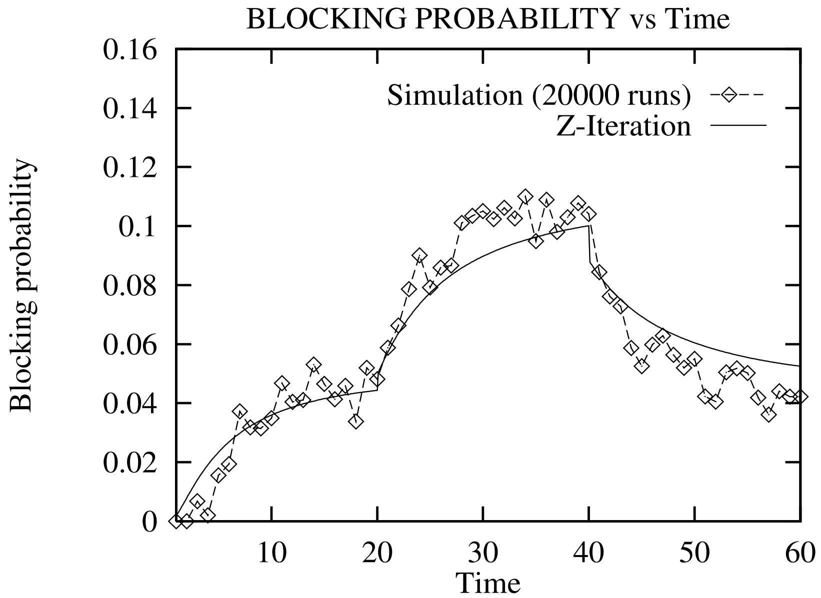

Figure 4.10: Blocking probability versus time. MCMR system with single-server resources. Time-varying arrivals.

Chapter 5

Application of the Z-Iteration to Detailed Network Models

In this chapter, we apply the Z-iteration to a detailed network model. We formulate the model in Section 5.1. Sections 5.2 and 5.3 illustrate how our method can capture the effects of various control schemes. Section 5.2 discusses scheduling and admission. Section 5.3 discusses routing. Section 5.4 investigates three routing schemes on the NSFNET backbone topology.

5.1 Network Model

We consider networks of arbitrary topology supporting real-time communication using a connection-oriented reservation scheme. That is, before a real-time application (e.g., voice, video) can start transmitting its packets at the requested end-to-end QoS (e.g., delay), a connection has to be first established along a fixed physical route from the source node to the destination node. For this, the source node uses its routing information to choose a potential route to the destination node.

A connection setup message is then sent over this route, requesting a local QoS from each of its links such that the aggregate of these local QoS satisfies the connection's end-to-end QoS. If the request fails at any link due to lack of resources (or any other admission constraints), the connection is blocked and lost; it is assumed that it is not attempted on another (alternate) route. Otherwise, the connection is established and resources are allocated to it. At the end of transmission, this connection is torn down and resources are released. We assume that a connection setup (and teardown) request on a multi-link route reaches all links of the route simultaneously.

Routing can be static or dynamic. For dynamic routing, we assume routing information is updated by periodic broadcasts by nodes of the status of their outgoing links during the last period. This periodic collection of status information is often used in routing algorithms proposed for integrated services networks (e.g., [1, 8, 24]). We assume that broadcasts of all nodes are synchronized; we can easily model unsynchronized broadcasts. We also assume that these broadcasts reach other nodes instantaneously; this is justifiable because the time to propagate routing information is small compared to the routing update period.

After each update, a node uses its new routing information to compute new routes to be used for incoming connections until the next broadcast. The routes are thus updated at discrete time instants $nT, n = 1, 2, \cdots$, where T is the routing update period.

Services

We think of the network as providing real-time services. A *service* represents connections with the same source-destination node pair and the same traffic and QoS parameters. The parameters of a service s include the following:

- Arrival rate of requests for a connection setup, $\lambda_s(t)$.
- Average lifetime of a connection from the time it is successfully established until it ends, $1/\mu_s(t)$.
- QoS requirements of a connection, for example, the end-to-end statistical delay bound $(D_s,\ \varepsilon_s)$ denoting that probability[end-to-end packet delay $> D_s$] $< \varepsilon_s$.
- Packet (or cell) generation characteristics of a connection, such as its mean transmission rate m_s and peak transmission rate M_s.

Classes

A connection of a service can potentially be established along any of the possible routes between the service's source node and the service's destination node. The *class* of a connection is defined by its service and the route it takes.

Figure 5.1 shows a network offering two services: service s1 from node 0 to node 4, and service s2 from node 1 to node 3. Each service has two possible routes for connection setup. Hence the network has four classes: classes c1 and c2 for s1 connections using route $\langle$ 0, 1, 2, 3, 4 $\rangle$ and $\langle$ 0, 5, 4 $\rangle$ respectively, and classes c3 and c4 for s2 connections using $\langle$ 1, 0, 5, 4, 3 $\rangle$ and $\langle$ 1, 2, 3 $\rangle$ respectively.

The instantaneous arrival rate of class-c connections of service s, denoted by $\lambda_c(t)$, is a function of $\lambda_s(t)$ and the routing algorithm. Note that with dynamic routing, class arrivals have time-varying statistics irrespective of whether the service arrivals have time-varying statistics.

Because a class is defined by the pair $\langle$ service, route $\rangle$, we can have a large number of classes, which may cause a computational bottleneck. To avoid this, we can restrict the set of possible routes, for example, to the shortest (in number of hops) and close to shortest paths.[1] This is

[1] Experiences with circuit-switched networks show that this restriction results in simple and efficient routing schemes [7, 79].

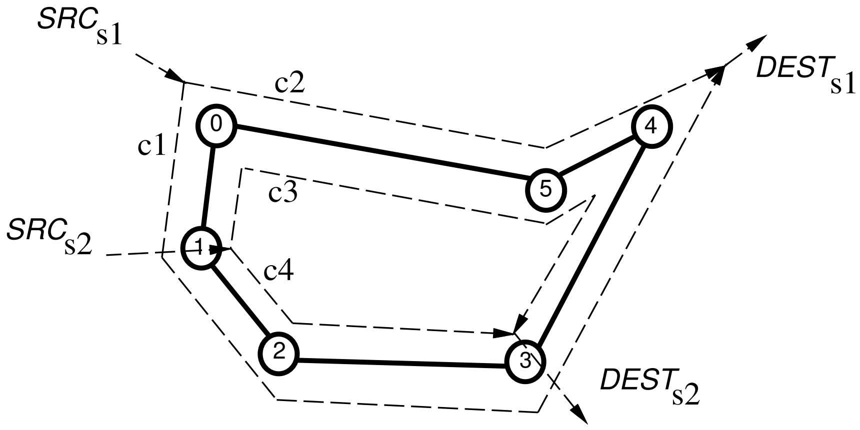

Figure 5.1: A network example.

acceptable because using a longer path for a connection ties up resources at more intermediate nodes, thereby decreasing network throughput. Furthermore, it also ties up more resources at each intermediate node because satisfying the end-to-end QoS requirement would require more stringent local QoS requirements. Section 5.3 addresses the selection of routes in more detail.

Obtaining class parameters at a link

Each link in the network is used by a subset of the classes. For example, in Figure 5.1, link $\langle 5, 4\rangle$ is used by two classes, namely c2 and c3. The parameters of a class at a link on its route are obtained from the parameters of its service. To do this, we make the following assumptions; some of these assumptions can be relaxed, possibly at additional computational cost:

- Connection setup requests arrive according to Poisson processes.
- The routing is probabilistic. That is, probabilities are assigned to the candidate paths and arriving connections are routed independently according to these path probabilities. With dynamic routing, the probabilities are periodically updated according to dynamic status information (e.g. measured load). Note that these probabilities could take the values 0 and 1 for single-path routing.
- For a connection setup request on a multi-link route, the requested end-to-end QoS is divided equally among the links. This is the so-called "equal allocation" policy. For example, if a connection of service s requesting an end-to-end QoS $(D_s,\ \varepsilon_s)$ is to be established on an h-link route, then we require that each link on the route guarantees a local requirement of $(\frac{D_s}{h},\ \frac{\varepsilon_s}{h})$ [85, 87].
- The packet generation characteristics of a connection established on a multi-link route do not change from link to link, i.e. remain the same as the given external characteristics.

The first assumption is often made and is reasonable in practice [41, 40, 90]. The second assumption uses a type of routing proposed in many studies (e.g., [8, 35]). The third assumption uses an end-to-end QoS allocation policy studied in [85, 45, 87].

The last assumption is valid in practice if the network admission control makes the same assumption, as for example, in the effective bandwidth approach by Guérin *et al.* [44]. It is also valid if the network uses a tightly-controlled approach that uses a non-work-conserving link scheduling discipline to reconstruct the traffic pattern at each link. An example of such approach is the Rate-Controlled-Static-Priority approach by Zhang and Ferrari [106]. Otherwise, the traffic pattern has to be characterized at each link as in [89, 25].

Given the above assumptions, it is straightforward to obtain the parameters of a class at a link. Consider, for example, the parameters of class c2 at link $r \in \mathcal{R}_{\mathrm{c2}} = \{\langle 0,\ 5\rangle,\ \langle 5,\ 4\rangle\}$. Connection setup requests arrive according to a Poisson process with rate $\lambda_{\mathrm{c2}}(t) = \alpha_{\mathrm{s1,c2}}(t)\ \lambda_{\mathrm{s1}}(t)$, where $\alpha_{\mathrm{s1,c2}}(t)$ is the (possibly dynamic) probability of a connection of service s1 being routed on class-c2 route. The average lifetime of a connection $\frac{1}{\mu_{\mathrm{c2}}(t)} = \frac{1}{\mu_{\mathrm{s1}}(t)}$. For an end-to-end QoS $(D_{\mathrm{s1}},\ \varepsilon_{\mathrm{s1}})$, the local QoS requirement $(D^r_{\mathrm{c2}},\ \varepsilon^r_{\mathrm{c2}}) = (\frac{D_{\mathrm{s1}}}{2},\ \frac{\varepsilon_{\mathrm{s1}}}{2})$, because the route of class c2 is two-hop long. The packet generation characteristics $(M_{\mathrm{c2}},\ m_{\mathrm{c2}},\ \cdots) = (M_{\mathrm{s1}},\ m_{\mathrm{s1}},\ \cdots)$.

The above model can be solved using the *Z*-iteration. The end-to-end measures of each service are easily obtained once the end-to-end measures of each of the service's classes are computed.

5.2 Scheduling and Admission

The *Z*-iteration accounts for scheduling at a link r through the set of feasible states $\mathcal{F}^r$. Recall that we define a state $(\sigma_1, \sigma_2, \cdots, \sigma_{|\mathcal{C}^r|})$ in $\mathcal{F}^r$ by the number of connections σ_c of each class $c \in \mathcal{C}^r$ that can be established simultaneously on link r, i.e. for which the local QoS is satisfied for every connection.

$\mathcal{F}^r$ can be determined using a packet-level analysis [44, 23] knowing the parameters of each class at link r (obtained as shown in Section 5.1) and the link scheduling algorithm. Note that $\mathcal{F}^r$ would typically be different for every link r because links have different capacities, are used by different sets of classes, etc. It is also different for different scheduling disciplines because disciplines resulting in looser performance bounds would typically have a smaller set of feasible states.

In the following, we illustrate the computation of $\mathcal{F}^r$ for a "per-connection" link scheduling algorithm of the weighted round-robin type. An example of this type of scheduling algorithms is weighted fair-queueing [89]. Here, each class-c connection is allocated (and guaranteed) a certain amount of bandwidth on link $r \in \mathcal{R}_c$ that is enough to satisfy its local QoS requirement. This required bandwidth depends of course on the local QoS and the packet generation characteristics of the connection.

Henceforth we assume that a connection of service s requests an end-to-end statistical delay bound $(D_s,\ \varepsilon_s)$, where the delay does not include the propagation delay. This QoS requirement is also referred to as packet jitter [33, 103]. This is typically required by applications such as voice since they can tolerate some packet loss (a packet is considered lost if its delay exceeds D_s) [32, 33].

If the connection is described by a two-state model where it is either in a busy state sending packets back-to-back at peak rate or in an idle state sending no packets at all, the required bandwidth[2], denoted by R_c^r, can be obtained from the following approximation derived in [6, 44, 31]:

$$R_c^r = M_c \frac{\beta_c^r - X_c^r + \sqrt{[\beta_c^r - X_c^r]^2 + 4\ X_c^r\ \rho_c\ \beta_c^r}}{2\ \beta_c^r} \tag{5.1}$$

where

- M_c is the peak rate of the connection.
- m_c is the mean rate of the connection.
- b_c is the average duration of the busy period.
- $\beta_c^r = ln(\frac{1}{\varepsilon_c^r})\ b_c\ (1 - \rho_c)\ M_c$.
- $\rho_c = \frac{m_c}{M_c}$ is the probability that the connection is active (in busy state).
- $X_c^r = D_c^r \times R_c^r$ is the buffer space required by the connection.

R_c^r can be computed from equation (5.1) iteratively. For each class $c \in \mathcal{C}^r$, we can then determine its requirements R_c^r and X_c^r. From this, we can determine whether a state $(\sigma_1, \sigma_2, \cdots, \sigma_{|\mathcal{C}^r|})$ belongs to $\mathcal{F}^r$; it must satisfy the following two conditions:

- $\sum_{c \in \mathcal{C}^r} \sigma_c\ R_c^r$ is no greater than the total capacity of link r, denoted by Cap^r.
- $\sum_{c \in \mathcal{C}^r} \sigma_c\ X_c^r$ is no greater than the total available buffer space of the link.

For ease of presentation, we assume that there is enough link buffer space such that the second condition is always satisfied. Then for a state to be feasible it suffices to only satisfy the first condition. Thus, Cap^r defines *r.max*, and R_c^r defines *c.r.req*.

We obtain $\mathcal{S}_c^r(.)$ by solving the Markov chain over $\mathcal{F}^r$. In particular, denoting by $P(\sigma)$ the probability of being in a state $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_{|\mathcal{C}^r|}) \in \mathcal{F}^r$, we have

$$P(\sigma) = P(0) \prod_{c'=1}^{|\mathcal{C}^r|} \frac{(\lambda_{c'}/\mu_{c'})^{\sigma_{c'}}}{\sigma_{c'}!} \tag{5.2}$$

where $P(0) = [\sum_{\sigma \in \mathcal{F}^r} \prod_{c'=1}^{|\mathcal{C}^r|} \frac{(\lambda_{c'}/\mu_{c'})^{\sigma_{c'}}}{\sigma_{c'}!}]^{-1}$ is the normalization constant. This solution is valid not only for exponentially distributed connection lifetimes [53], but also for generally-distributed lifetimes [55].

[2] Often referred to as effective or equivalent capacity [31, 59, 6, 44, 1].

Assuming a simple admission control where the arrival of a new class-c connection is blocked if its admission would lead to a nonfeasible state, we have

$$B_c^r = \sum_{\sigma \in \mathcal{F}^r} I\{(\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^r|}) \notin \mathcal{F}^r\} \; P(\sigma) \tag{5.3}$$

where

$$I\{(\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^r|}) \notin \mathcal{F}^r\} = \begin{cases} 1 & \text{if } (\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^r|}) \notin \mathcal{F}^r \\ 0 & \text{otherwise} \end{cases}$$

This is often referred to as "complete-sharing" admission control [50]. Note that $I\{.\}$ defines the set of blocking states. Other admission control schemes can be modeled by alternative definitions of $I\{.\}$.

The computation of $\mathcal{F}^r$ is typically expensive as it requires determining the $|\ \mathcal{C}^r\ |$-dimension feasible states [74, 70]. In addition, given the admission control policy, we need to determine for each class which of the feasible states are blocking. This computational complexity is reduced if we assume $\{R_c^r : \ c \in \mathcal{C}^r\}$ and Cap^r are integers and view the link state as belonging to the set $\{0, 1, 2, \ldots, Cap^r - 1, Cap^r\}$, where the state indicates the amount of bandwidth reserved. This *one-dimensional* link model has a simple steady-state solution in the multi-rate circuit switching literature [92], and was given in Section 3.1.[3] Note that here $\mathcal{F}^r$ is implicitly defined by the constraint on state j satisfying $0 \leq j \leq Cap^r$. This link model is usually referred to as the *stochastic Knapsack* model [20, 21]. The function $\mathcal{T}_c^r(.)$ is defined by N_c^r as in Section 3.1.

5.3 Routing

The Z-iteration accounts for routing through the time-dependent class arrival rates $\lambda_c(t)$. These are affected in our model by the route selection probabilities $\alpha_{s,c}(t)$. We assume the $\alpha_{s,c}(t)$ are periodically computed based on the network topology and load averaged over the last period. The load information consists of link/path measurements, which may include quantities such as reserved link capacity and path blocking probability. Obviously these quantities should be measurable in practice; indeed a node can measure the reserved capacity for each of its outgoing links from the connection setup/teardown procedure. Also, a source node can measure the blocking probability of a path if we assume that when a setup fails at an intermediate node, this node sends a "reject" message back to the source.

These quantities should also be obtainable from our model. We can obtain the average reserved link capacity from the average number of established connections and the effective capacity of each

[3] In a multi-rate circuit-switched network, each call may request a different number of channels. This number is however the *same* on every link along *any* route the call might take. This is not the case in the networks we are considering where the bandwidth required by a connection on a link depends on the number of links along the route taken by the connection.

of the link's classes, which we compute in our model. We can also obtain a path blocking probability from the classes' blocking probabilities, which we also compute in our model.

We are interested in route selection algorithms for networks of *arbitrary* topologies and offering *heterogeneous* services. We want algorithms that result in low blocking probabilities (a high successful setup rate) and hence high network throughput. Our model can capture several design choices when developing such algorithm. One design choice is related to the set of candidate paths the source node would consider for connection routing. This determines the number of classes defined for each service. We do not want the source node to consider paths that are too long since this would result in increased utilization and hence reduced throughput. So the set of candidate paths could consist of only minimum-hop paths, or it could consist of both minimum-hop paths and next-to-minimum-hop paths. (By a next-to-minimum-hop path, we mean a minimum-hop + i path for the smallest $i \in \{1, 2, \cdots\}$ such that a path exists.)

Routing schemes designed for circuit-switched networks [40] and recently proposed for ATM networks [45, 48, 46, 47] consider one-hop and two-hop paths only. Routing schemes that consider paths of arbitrary hop length are often proposed for the Internet [96, 17]. Our model can evaluate both types of schemes.

From the set of candidate paths, we should determine which path to use for routing the setup request message for a new incoming connection. A path p could be selected probabilistically at random or using path weights W_p where[4]

$$W_p \propto \frac{F_p}{H_p \times L_p} \tag{5.4}$$

where H_p is the number of hops of path p (this gives preference to shortest paths), L_p is a measure of the load on path p averaged over the last update period (discussed below), and F_p is either 1 or 0 depending on whether the path p is feasible or not; a path p is said to be feasible if the source "expects" a successful setup on p [1].[5] The $\alpha_{s,c}(t)$ can then be computed according to (5.4).

Another design issue is related to how L_p is defined. For example, L_p could be (i) the blocking probability of path p, (ii) the sum of the utilizations of the links on path p, where the utilization of a link is the fraction of the link capacity reserved, (iii) the maximum link utilization of the links on path p, or (iv) the sum of the delays of the links on path p, where the delay of a link r can be estimated as $\frac{1}{Cap^r - CapRes^r}$ where $CapRes^r$ is the average reserved link capacity (note that this delay estimation uses the $M/M/1$ delay formula [63]).

[4] W_p may depend on other factors. We use here the ones that were considered in previous works (e.g., [8, 1, 17]) when selecting routes for connections.

[5] The source would take into account the requirements of the new connection in addition to the current load on the path (assuming it is accurate) to test the feasibility of the path. This is in fact an admission control function.

5.4 Numerical Results for NSFNET

In this section, we use our model to compare three route selection algorithms. We assume the use of the "per-connection" link scheduling and complete-sharing admission described in Section 5.2. The required bandwidths R_c^r are computed using equation (5.1); if the computed value is not integer, it is rounded to the smallest integer greater than this value. We assume adequate buffer space.

We consider the performance of the routing algorithms on the topology of the NSFNET backbone shown in Figure 5.2. All links have capacities of 600. The time step δ equals 0.1. The routing update period T equals 5. We consider 52 services using the NSFNET backbone, with parameters as shown in Figure 5.1. Services with the same traffic and end-to-end QoS parameters, but with different source/destination pairs, are grouped in the same row.

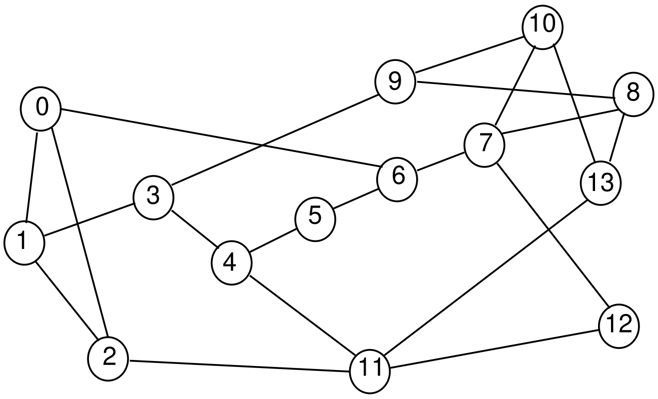

Figure 5.2: NSFNET backbone: 14 nodes, 21 bidirectional links, average degree 3.

We assume a source node considers only the set of minimum-hop and minimum-hop + 1 paths for connection routing. A path from the set is selected probabilistically according to path weights as explained in Section 5.3. The first selection algorithm, referred to as SEL.HOP, defines the path weight as $1/H_p$. The second selection algorithm, referred to as SEL.UTIL, defines the path weight as $(1 - U_p)$, where U_p is the maximum link utilization of the links on path p. The third selection algorithm, referred to as SEL.UTIL_HOP, defines the path weight as $(1 - U_p)/H_p$.

Figure 5.3 shows the instantaneous network throughput for the three routing algorithms. Figure 5.4 shows their instantaneous blocking probabilities. We observe that SEL.UTIL_HOP performs the best, closely followed by SEL.UTIL, and then by SEL.HOP, which is much worse. Clearly, for this network configuration, choosing paths which are both under-utilized and short for routing new incoming connections is the best strategy. We note that this is consistent with results in [8] where a route selection algorithm similar to SEL.UTIL_HOP was shown to outperform other algorithms on a 5-node connection-oriented reservationless network using discrete-event simulations. To obtain a curve here, the Z-iteration required around 45 minutes of execution time rather than the tens of hours that simulation would have required.

$(SRC_s,\ DEST_s)$	$(M_s,\ m_s,\ b_s,\ D_s,\ \varepsilon_s)$	$(\lambda_s,\ \mu_s)$
(0, 13),(1, 13),(2, 13),(3, 13),(4, 13),(5, 13)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(0, 13),(1, 13),(2, 13),(3, 13),(4, 13),(5, 13)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(6, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(2, 2)
(6, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(2, 2)
(7, 13),(8, 13),(9, 13),(10, 13),(11, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
(7, 13),(8, 13),(9, 13),(10, 13),(11, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
(12, 13)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)
(12, 13)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)
(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(2, 1),(3, 1),(4, 1),(5, 1),(6, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(7, 1),(8, 1),(9, 1),(10, 1),(11, 1),(12, 1),(13, 1)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(2, 1),(3, 1),(4, 1),(5, 1),(6, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
(7, 1),(8, 1),(9, 1),(10, 1),(11, 1),(12, 1),(13, 1)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)

Table 5.1: Parameters of the 52 services using the NSFNET backbone.

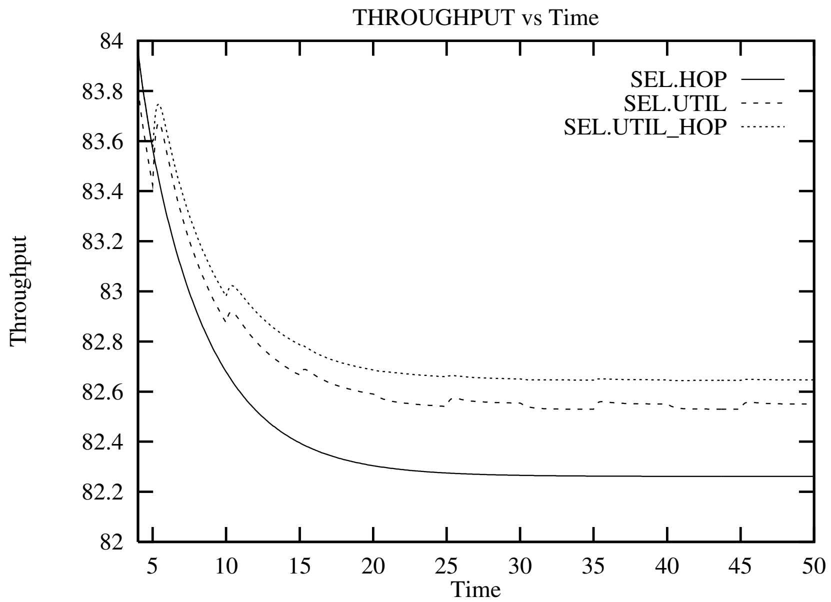

Figure 5.3: Total throughput versus time for the NSFNET backbone.

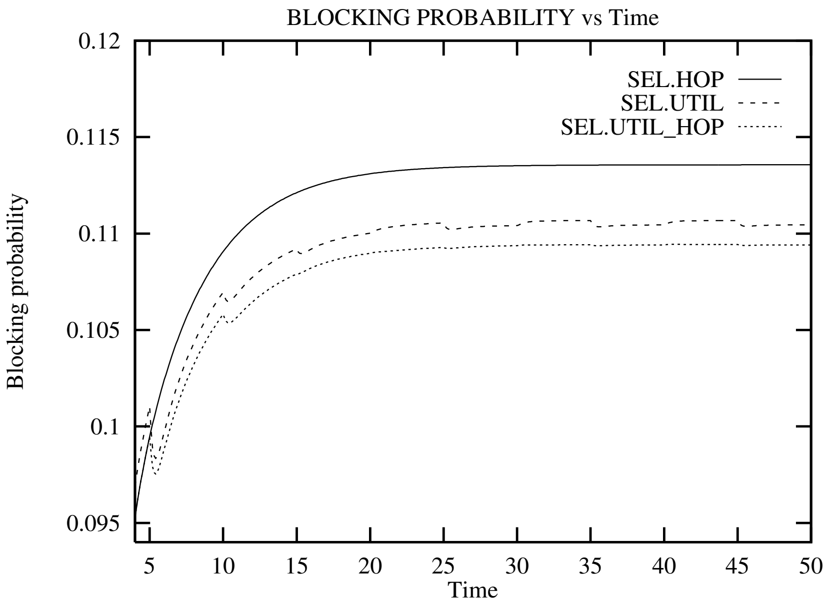

Figure 5.4: Blocking probability versus time for the NSFNET backbone.

Chapter 6

Quasi-Static Evaluation of a Type-of-Service Datagram Network

6.1 Introduction

With the increasing diversity of network applications, it has become crucial for networks, such as the Internet, to offer various services, including best-effort services and guaranteed services. A guaranteed service provides bounds on performance. A best-effort service can provide qualitatively better service, but without the quantitative bounds of a guaranteed service [93]. The Internet Protocol (IP) currently provides only best-effort services. Extensive effort is underway to extend IP to support other services [16, 22, 93].

The focus of this and the next chapter is the provision of different *type-of-service* (TOS) classes of best-effort service. To offer various TOS requirements, e.g. low delay or high throughput, a network's routing protocol should be able to determine appropriate routes for each TOS class. We are concerned here with next-hop (or datagram) routing, because it has proved to be a simple and robust way to do adaptive routing of best-effort traffic [78, 60]. Source (or virtual-circuit) routing on the other hand is often used for guaranteed services.

Next-hop TOS routing is done as follows. Each node maintains for each destination node and TOS class, a neighboring node id, referred to as the *next-hop*. Every data packet header contains its destination id and the TOS class of the application. When a node receives a data packet, it forwards the packet to the next-hop for the packet's destination and TOS class. The objective of the routing protocol is to choose next-hops so that the resulting routes satisfy the requested type of service. The quality of service offered by a route depends on the traffic through its links, which depends on the time-varying external load. Consequently, a routing protocol must monitor link traffic changes and adapt its next-hops. To do this, each node maintains for each outgoing link and TOS class, a dynamic *link cost*, which is updated regularly according to the traffic flowing through the link. This link cost information is regularly disseminated to nodes of the network. Based on received link cost information, each node maintains and regularly updates its next-hop for each destination and TOS class.

The IP layer of the Internet Protocol suite specifies different TOS classes [5]. Among them are the minimum delay service required for example by interactive traffic or real-time traffic, and the maximum throughput service required for example by bulk transfers such as network mail or FTP. Routing protocols such as the Internet OSPF [83] and the OSI IS-IS [18] provide separate next-hops for each TOS class. However, the TOS mechanism has been so far of little use, and little is known on how well it would work in practice. In addition, many current routing protocols use *static* link costs, typically configured by the network administrator and responding only to failures and recoveries.

To our knowledge, only one approach to adaptive TOS routing has been proposed [38]. This approach, henceforth called TOS1, considers two TOS classes: *low delay* and *high throughput*. We refer to traffic of the former class as *delay-sensitive* traffic, and of the latter class as *throughput-sensitive* traffic. TOS1 uses measured link delays as the link costs for the delay-sensitive traffic (delay-based routing). It uses link utilizations, or equivalently available link capacities, as the link costs for the throughput-sensitive traffic (utilization-based routing). In TOS1, each node maintains for each outgoing link, a *single* FCFS queue of data packets; that is, packets of every TOS class share this queue. Reference [38] refers to simulation studies but does not present any quantitative results.

Traditionally, link queueing has been the FCFS discipline. It appears desirable to use a more structured queueing discipline that helps "isolate" the different TOS classes, for example, by using a separate queue for each TOS class. This concept of *isolating* traffic classes using structured queueing disciplines has been used recently in flow control studies, e.g. [26, 58, 14, 89, 34]. Here, we investigate the use of a structured queueing discipline with adaptive next-hop TOS routing.

Our approach

We consider the following simple link scheduling discipline, henceforth referred to as TOS queueing. We consider two TOS classes: low delay and high throughput. Each node maintains two FCFS queues for each outgoing link, one for each TOS class. The link bandwidth is allocated equally between the two queues in a round-robin fashion. (This is similar to the fair-queueing discipline [26], except that the link bandwidth is divided equally amongst the TOS classes rather than the connections using the link.)

For any link, the link cost for delay-sensitive traffic is obtained by exponentially averaging the measured delay that is experienced by delay-sensitive packets *only*. The link cost for throughput-sensitive traffic is obtained by exponentially averaging the measured utilization of the link, i.e. accounting for *both* delay-sensitive and throughput-sensitive packets. Henceforth, we refer to our approach as TOS2.

Our discrete-event simulations on a subset of the NSFNET-T1-Backbone topology show that

TOS2 performs significantly better than TOS1 in a typical situation [52] where the proportion of delay-sensitive traffic is small compared to the throughput-sensitive traffic. As expected, TOS2 achieves a lower end-to-end delay for delay-sensitive packets since it effectively gives them higher priority. Unexpectedly, TOS2 also yields a lower overall end-to-end delay.

We argue that this is because TOS2 achieves significantly improved routing by exploiting the scheduling structure of TOS queueing when calculating link costs. In particular, the routes of the two traffic classes can be *isolated* with the delay-sensitive traffic taking the low delay routes, and the throughput-sensitive traffic taking the under-utilized routes. This results in a better overall network performance.

In fact, we find that a non-TOS scheme, which does not distinguish between the two types of traffic and applies utilization link cost to both, referred to as UTIL, performs significantly better than TOS1 at high load.

To gain more insight into the system behaviors of TOS1 and TOS2, we analyze a simple quasi-static model of a single source-destination node pair connected by two parallel paths, the first path representing low delay routes and the second path representing high capacity routes. We view this system as a dynamical system [10]. We represent isolation by a *stable* state where all delay-sensitive traffic stays on the first path and all throughput-sensitive traffic stays on the second path. We apply the Liapunov function method to derive stability theorems. We show that for certain parameter values, the isolation state provides the best delay performance for both traffic classes. We also show that TOS2 has a larger stability region corresponding to isolation than TOS1. Starting at a state outside this stability region would lead to simultaneous route oscillations for both traffic classes resulting in a bad delay performance.

Section 6.2 describes our discrete-event simulation model and results. Section 6.3 gives our quasi-static model. The stability analysis of this model is presented in Chapter 7. Appendix A describes details of the simulations, including performance measures, scenarios, and plots. Appendix B contains details of a derivation.

6.2 Discrete-Event Simulations

Our simulation studies were done with a discrete-event simulator, MaRS [3], which has been used for other studies of routing algorithms [95, 94]. Subsection 6.2.1 describes the simulation model. Subsection 6.2.2 presents general observations about the results.

6.2.1 Model

Regarding the physical network, we consider the "East coast" subset of the NSFNET-T1-Backbone. Figure 6.1 illustrates the topology. Link propagation delays in milliseconds are indicated.

We have two versions: a low-speed version (with NSFNET-T1 parameters) and a high-speed

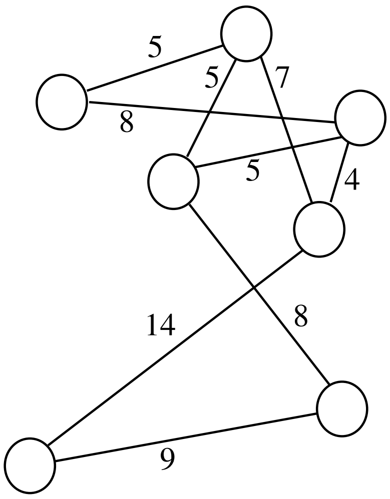

Figure 6.1: The "East coast" subset of the NSFNET-T1-Backbone (7 nodes, 9 bidirectional links).

version. There are no link or node failures. All nodes have adequate buffer space for buffering packets awaiting processing and forwarding.

Regarding link scheduling, we consider two disciplines for the scheduling of data packets over the links: FCFS and TOS queueing. Both TOS1 and UTIL use FCFS queueing. TOS2 uses TOS queueing. In all schemes, routing packets have priority over data packets; i.e. data packets can be scheduled for transmission only if there are no routing packets present.

Regarding routing, we consider a link-state algorithm like SPF (Shortest Path First) used in the ARPANET [78] and OSPF (Open SPF) used in the Internet [83]. Each node maintains a time-varying cost (explained below) for each outgoing link and TOS class. Each node also maintains a view of the network topology, with a cost for each TOS class and link in the network. To keep these views up-to-date, each node regularly broadcasts the link costs of its outgoing links to all other nodes using flooding. As a node receives this information, it updates its view of the network topology and applies Dijkstra's shortest path algorithm [28] to choose its next-hop for each destination and TOS class. (Using a more scalable mechanism to disseminate link costs would not affect our conclusions.)

The method used to compute link costs for each TOS class depends on the link scheduling discipline. In all cases, each node's outgoing link costs are updated regularly. A link cost is always a simple moving average of a "raw cost", which is some measure of current link traffic.

In TOS1 and TOS2, each node maintains the following two raw-costs for each outgoing link:

- *RawUtilization*: percentage of time the communication channel is busy transmitting a packet; and
- *RawDelay*: In TOS1, this is the average packet delay (queueing, transmission, and propagation) in milliseconds as experienced by all data packets. In TOS2, this is the average delay

in milliseconds as experienced by delay-sensitive packets only.

Let $LinkCost(D)$ and $LinkCost(T)$ denote the link cost for delay-sensitive and throughput-sensitive traffic, respectively. Then at the end of each update interval, they are updated as follows:

$$\begin{aligned} LinkCost(D) &:= b \times RawDelay + (1-b) \times LinkCost(D) \\ LinkCost(T) &:= b \times RawUtilization + (1-b) \times LinkCost(T) \end{aligned}$$

where the constant b satisfies $0 < b < 1$.

Recall that UTIL does not use any TOS facility. The utilization metric is used to compute one next-hop for both TOS classes, i.e. $LinkCost(T)$ is used for all traffic.

With a utilization-based link cost metric, it seems natural to define the cost of a path as the minimum available link bandwidth (or equivalently, highest link utilization) of the links along the path. However, we found that such a path-cost metric leads to large routing oscillations and instability even at low workload. Therefore, we set the path-cost metric to the sum of the link costs along the path from the source node to the destination node and use Dijkstra's shortest path algorithm as in [42]. (For each of these two path-cost metrics, it is easy to come up with static scenarios where it outperforms the other.)

Regarding workload, this is defined in terms of $\langle$source node, destination node$\rangle$ pairs. In each pair, the source produces data packets to be delivered to the destination. A source produces data packets according to a packet-train model [51]. The workload consists of two parts, a delay-sensitive workload and a throughput-sensitive workload. For both parts, we use a uniform distribution of source-destination pairs over the nodes of the network. Let parameter $U(D)$ ($U(T)$) denote the average number of source-destination pairs between every two nodes for delay-sensitive (throughput-sensitive) traffic. We have also investigated skewed distribution of source-destination pairs and obtained similar results.

6.2.2 Observations

In this subsection, we present general observations about the simulation results. Detailed descriptions of scenarios simulated and plots of the observed performance measures are given in Appendix A.

In every scenario, the system behaves in a manner typical of open queueing networks [63]. That is, the throughput equals the workload as long as the workload is less than the system capacity; for workload higher than the system capacity the system is unstable. With increasing workload, the delay increases at first slowly until a point where the system starts becoming saturated; we refer to this point as the saturation point. Further increase in the workload beyond this point causes the delay to increase dramatically (with increasing rate) until the system becomes unstable.

Fixing $U(D)$ and varying $U(T)$ in a range where the delay-sensitive traffic constitutes almost 25%-30% of the total traffic, we found that TOS2 performs significantly better than TOS1 with respect to delays; TOS1 reaches saturation sooner. See Figure 6.2.

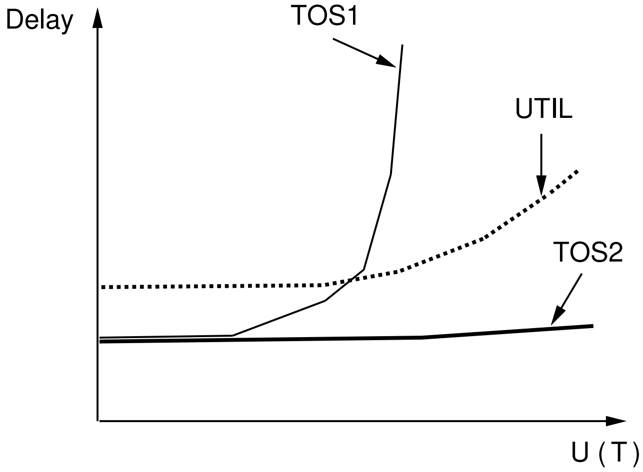

Figure 6.2: A generic plot. Delay versus $U(T)$ for a fixed $U(D)$.

Our explanation is as follows: It is well known that delay-based routing does not perform well at high load when queueing delay is a significant part of measured link delay, which consists of queueing, transmission, and propagation delays [60, 42, 13]. This is mainly because from the classical delay-utilization curve, around saturation, a small increase in utilization corresponds to a large increase in link delay. This dramatic change can result in the link becoming unattractive and thus being avoided by *all* delay-sensitive sources. Consequently, at the next routing update the link reports a very low cost and becomes attractive again. This leads to oscillatory behavior, which in turn degrades performance [60]. This is the case with TOS1 due to the use of the FCFS link scheduling discipline.

In TOS2, because delay-sensitive packets have a lower queueing delay under type-of-service queueing [26], the measured link delay becomes dominated by transmission and propagation delays. Thus, the reported delay link costs do not change dramatically and the delay metric remains a good indicator of expected link delay after updating the routes [60]. This improves the performance of delay-based routing of delay-sensitive packets and results in more stable routes for that traffic class. Meanwhile, the link utilization metric makes the throughput-sensitive traffic move away from the delay-sensitive traffic, taking the under-utilized routes. This has the effect of *isolating* the two traffic classes, resulting in a better overall network performance.

Intuitively, isolation is desirable since otherwise it becomes more likely that both traffic classes will move away from a highly loaded link (i.e. with high delay and utilization) at the same routing update. Such simultaneous traffic shifts degrade the overall performance, and in particular result in a higher delay and network under-utilization.

Ignoring low values of $U(T)$, UTIL also provides lower delays than TOS1. However, UTIL performs worse than TOS2 over the whole range of $U(T)$. This is expected, since the utilization-based metric does not necessarily result in minimum delay routes, especially at light load [60].

6.3 Quasi-Static Model

In this section, we consider a simple network model to gain more insight into the complicated behaviors of TOS1 and TOS2. We analyze the model in Chapter 7.

We model a network by a source node sending traffic to a destination node along two paths. Path 1 represents low delay routes, and path 2 represents high capacity routes. Path i $(i = 1, 2)$ has propagation delay P_i time units and average transmission capacity C_i packets/time unit. There are N delay-sensitive connections, and M throughput-sensitive connections from the source node to the destination node. For every connection, packets originate at the source node according to a Poisson process, and without loss of generality we assume an arrival rate of 1 packet/time unit (alternatively we could think of the x's and y's below as the total arrival rates).

At any instant, we describe the state of the network by the tuple (x, y), where x is the number of delay-sensitive connections on path 1, and y is the number of throughput-sensitive connections on path 1. To model routing updates, we use a discrete-time flow approach as in [12]. We assume that (some or all) connections periodically update their routes to the destination node every $\triangle$ time units, where $\triangle$ is long enough for the network to reach steady-state after a routing update. Routes, and hence the network state, are updated at discrete time instants $(k+1)\triangle$, $k = 0, 1, 2, \cdots$. Let (x_k, y_k) be the network state immediately after time $k\triangle$. At an update instant $(k+1)\triangle$, we use steady-state $M/M/1$ results [63] to estimate link costs based on (x_k, y_k). Using these link costs, routes are updated and consequently the new network state (x_{k+1}, y_{k+1}) is obtained.

We denote by $T_{i,k+1}$ and $\rho_{i,k+1}$ the delay and utilization cost of path i, respectively, at time $(k+1)\triangle$. Recall that for an $M/M/1$ queue with offered flow f and service rate μ $(> f)$, the delay (queueing + service) equals $1/(\mu - f)$ and the utilization equals f/μ.

For TOS1, with a FCFS discipline at the source node, we can write the delay link costs as follows:

$$\begin{aligned} T_{1,k+1} &= \frac{1}{C_1 - (x_k + y_k)} + P_1 \\ T_{2,k+1} &= \frac{1}{C_2 - (\bar{x}_k + \bar{y}_k)} + P_2 \end{aligned} \tag{6.1}$$

where $\bar{x}_k = N - x_k$ and $\bar{y}_k = M - y_k$ denote the number of delay-sensitive connections and the number of throughput-sensitive connections, respectively, on path 2.

The utilization link costs are

$$
\begin{aligned}
\rho_{1,k+1} &= \frac{x_k + y_k}{C_1} \\
\rho_{2,k+1} &= \frac{\bar{x}_k + \bar{y}_k}{C_2}
\end{aligned}
\tag{6.2}
$$

The network state is updated using the costs of the two paths as follows:

$$
\begin{aligned}
x_{k+1} &= \begin{cases} (1-\alpha_k)\, x_k & \text{if } T_{2,k+1} < T_{1,k+1} \\ x_k + \alpha_k\, \bar{x}_k & \text{otherwise} \end{cases} \\
y_{k+1} &= \begin{cases} (1-\alpha_k)\, y_k & \text{if } \rho_{2,k+1} \leq \rho_{1,k+1} \\ y_k + \alpha_k\, \bar{y}_k & \text{otherwise} \end{cases}
\end{aligned}
\tag{6.3}
$$

The parameter α_k $(0 < \alpha_k \leq 1)$ reflects the amount of traffic rerouted. It can also be thought of as the degree of routing update synchronization at different nodes. Unless otherwise indicated, we assume that α_k is uniformly distributed over $[\alpha_{MIN}, \alpha_{MAX}]$, where $\alpha_{MAX} - \alpha_{MIN} = 0.2$, and $0.1 < \frac{\alpha_{MAX}+\alpha_{MIN}}{2} \leq 0.9$.

For TOS2, with TOS queueing at the source node, the two queues at an output link are correlated, which makes the analysis difficult. A number of approximate solutions for such systems, referred to as 1-limited polling systems, have been proposed. (See [99] for a good survey.) One common approach is to approximate the system by two loosely-coupled $M/M/1$ queues [81, 108]. The service rate of each queue depends on the utilization of the other queue.

Define $C^{eff}_{i,k}$ as the effective capacity available for the delay-sensitive traffic on path i after time $k\triangle$. We have

$$
C^{eff}_{i,k} \geq 0.5 C_i \tag{6.4}
$$

with the worst case occurring when the other queue is *always* not empty. Assuming each queue is $M/M/1$, we obtain the following (details in Appendix B):

$$
\begin{aligned}
C^{eff}_{1,k} &= \frac{(C_1 - 0.5(y_k - x_k)) + \sqrt{(C_1 - 0.5(y_k - x_k))^2 - 2C_1 x_k}}{2} \\
C^{eff}_{2,k} &= \frac{(C_2 - 0.5(\bar{y}_k - \bar{x}_k)) + \sqrt{(C_2 - 0.5(\bar{y}_k - \bar{x}_k))^2 - 2C_2 \bar{x}_k}}{2}
\end{aligned}
\tag{6.5}
$$

Thus we have the following delay link costs with type-of-service queueing:

$$
\begin{aligned}
T_{1,k+1} &= \frac{1}{C^{eff}_{1,k} - x_k} + P_1 \\
T_{2,k+1} &= \frac{1}{C^{eff}_{2,k} - \bar{x}_k} + P_2
\end{aligned}
\tag{6.6}
$$

The utilization link costs are as defined in (6.2). The network state is updated as in (6.3).

We refer to the iteration defined in (6.3) as I, i.e. $(x_{k+1}, y_{k+1}) = I(x_k, y_k)$. I is a mapping from a set G into itself, where $G = \{(x,y) : 0 \leq x \leq N \wedge 0 \leq y \leq M\}$. The sequence of points (x_1, y_1), (x_2, y_2), $\ldots$ is called the *trajectory* of the system [65, 86]. The trajectory may or may not converge to a fixed point (x^*, y^*), i.e. $(x^*, y^*) = I(x^*, y^*)$. The convergence to a fixed point indicates that the system stabilizes into a particular routing pattern, i.e. a particular amount of traffic routed for each class on each path. On the other hand, non-convergence indicates that the system oscillates between different routing patterns (or has chaotic behavior). Note that I is not a continuous mapping, and well-known theorems for convergence requiring this property cannot be directly applied [61].

We represent *isolation* by a stable state where every delay-sensitive connection stays on path 1, and every throughput-sensitive connection stays on path 2. This is equivalent to say that our iterative method converges to the fixed point $(N, 0)$. In the next chapter, we first derive sufficient conditions for the system to reach isolation as a function of the starting state. (In a real network, the starting state would be the result of arrivals of new connections, departures of old connections, failure/recovery of links, etc.) We then obtain less refined sufficient conditions for isolation, independent of the starting state.

Chapter 7

Analysis of Quasi-Static Model

In this chapter, we analyze the quasi-static analytical model presented in Section 6.3. In Section 7.1, we apply the Liapunov function method to derive stability conditions for the routes of the two traffic classes under both TOS1 and TOS2. We conclude with some remarks in Section 7.2. Appendix C contains details of proofs.

7.1 Stability Analysis

We use the Liapunov function method [86] to obtain sufficient conditions for stability and convergence to a fixed point without actually solving the system equations. The basic idea is to find a positive-definite scalar function $V(S)$, where S is the system state, such that its forward difference $\triangle V(S)$ taken along a trajectory is always negative. $V(S)$ is said to be a Liapunov function, and is regarded as a measure of the distance of the state S from the fixed point. As time increases, $V(S)$ decreases and finally shrinks to zero, i.e. the fixed point is approached.

It is more convenient to deal with the fixed point $(0,0)$ rather than $(N,0)$. Thus, we define the network state by $(\bar{x},y)$ instead of (x,y). Then, the iteration I defined in (6.3) becomes:

$$\bar{x}_{k+1} = \begin{cases} (1-\alpha_k)\,\bar{x}_k & \text{if } T_{1,k+1} \leq T_{2,k+1} \\ \bar{x}_k + \alpha_k\, x_k & \text{otherwise} \end{cases}$$
$$y_{k+1} = \begin{cases} (1-\alpha_k)\,y_k & \text{if } \rho_{2,k+1} \leq \rho_{1,k+1} \\ y_k + \alpha_k\, \bar{y}_k & \text{otherwise} \end{cases} \tag{7.1}$$

Combining (6.1), (6.2), and (7.1), the system behavior with TOS1 is described by the following:

$$\begin{aligned} \bar{x}_{k+1} &= (1-\alpha_k)\,\bar{x}_k + \alpha_k\, N\, \delta_k \\ y_{k+1} &= (1-\alpha_k)\,y_k + \alpha_k\, M\, \beta_k \end{aligned} \tag{7.2}$$

where

$$\delta_k = \begin{cases} 0 & \frac{1}{C_1-N+(\bar{x}_k-y_k)} + P_1 \leq \frac{1}{C_2-M-(\bar{x}_k-y_k)} + P_2 \\ 1 & \text{otherwise} \end{cases}$$

$$\beta_k = \begin{cases} 0 & \frac{M+(\bar{x}_k-y_k)}{C_2} \leq \frac{N-(\bar{x}_k-y_k)}{C_1} \\ 1 & \text{otherwise} \end{cases} \tag{7.3}$$

At the fixed point (which is now the origin), $\bar{x}_k \rightarrow 0, \bar{x}_{k+1} \rightarrow 0, y_k \rightarrow 0$, and $y_{k+1} \rightarrow 0$. Consequently, for the equations (7.2) to be satisfied, $\delta_k \rightarrow 0$ and $\beta_k \rightarrow 0$, which imply the following necessary (but not sufficient) conditions for convergence to the origin:

$$\begin{aligned} \frac{1}{C_1 - N} + P_1 &\leq \frac{1}{C_2 - M} + P_2 \\ \frac{M}{C_2} &\leq \frac{N}{C_1} \end{aligned} \tag{7.4}$$

Define $D_1 = \{(\bar{x}, y) : \frac{1}{C_1-N+(\bar{x}-y)} + P_1 \leq \frac{1}{C_2-M-(\bar{x}-y)} + P_2 \ \wedge\ \frac{M+(\bar{x}-y)}{C_2} \leq \frac{N-(\bar{x}-y)}{C_1} \ \wedge\ 0 \leq \bar{x} \leq N \ \wedge\ 0 \leq y \leq M\}$. See Figure 7.1.[1]

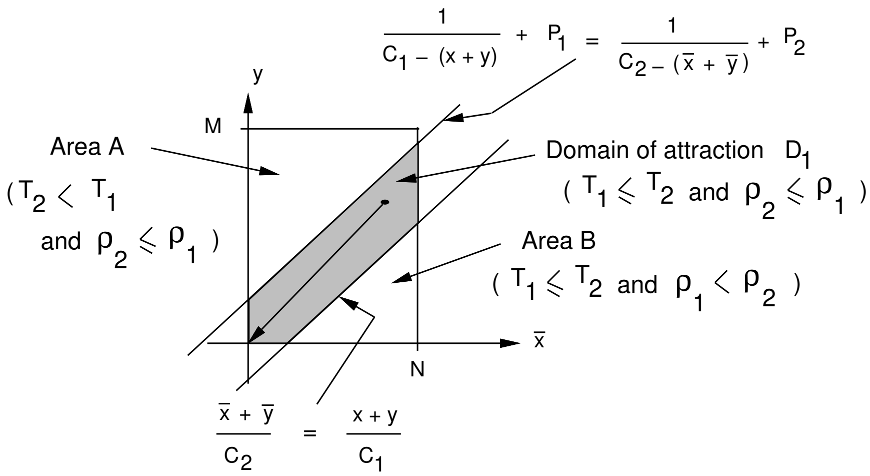

Figure 7.1: Domain of attraction for TOS1.

Define the scalar function $V(\bar{x}, y)$:

$$V(\bar{x}, y) = \bar{x}^2 + y^2 \tag{7.5}$$

[1]Although the equation of the left boundary of the domain D_1 is quadratic in x and y, the resulting solution turns out to be a straight line because one of the roots is infeasible. This is confirmed by monte-carlo simulations. See numerical example in Subsection 7.1.1.

We show that $V(\bar{x}, y)$ is a Liapunov function in D_1, which implies that starting from any point in D_1, the trajectory stays inside D_1, and converges to the origin.

Lemma 7.1.1 *$V(\bar{x}, y)$ is a Liapunov function in D_1. That is: $V(\bar{x}, y)$ is positive definite; and for all $(\bar{x}_k, y_k) \epsilon D_1 - \{(0,0)\}$, $\triangle V(\bar{x}_k, y_k) < 0$ and $(\bar{x}_{k+1}, y_{k+1}) \epsilon D_1$.* □

Proof. Since $V(\bar{x}, y) > 0$ for all $(\bar{x}, y) \neq (0,0)$ and $V(0,0) = 0$, then $V(\bar{x}, y)$ is positive definite.

The forward difference $\triangle V(\bar{x}_k, y_k)$ is computed as follows.

$$\begin{aligned}
\triangle V(\bar{x}_k, y_k) &= V(\bar{x}_{k+1}, y_{k+1}) - V(\bar{x}_k, y_k) \\
&= ((1-\alpha_k)\, \bar{x}_k + \alpha_k\, N\, \delta_k)^2 + \\
&\quad ((1-\alpha_k)\, y_k + \alpha_k\, M\, \beta_k)^2 - \\
&\quad (\bar{x}_k^2 + y_k^2) \\
&= -[1-(1-\alpha_k)^2]\, \bar{x}_k^2 - \\
&\quad [1-(1-\alpha_k)^2]\, y_k^2 + \\
&\quad (\alpha_k\, N\, \delta_k)^2 + (\alpha_k\, M\, \beta_k)^2 + \\
&\quad 2(1-\alpha_k)\, \alpha_k\, N\, \bar{x}_k\, \delta_k + \\
&\quad 2(1-\alpha_k)\, \alpha_k\, M\, y_k\, \beta_k
\end{aligned} \tag{7.6}$$

Consider a point $(\bar{x}_k, y_k) \epsilon D_1 - \{(0,0)\}$. Then $\delta_k = \beta_k = 0$. From equation (7.6), since $0 < [1-(1-\alpha_k)^2] \leq 1$, we have $\triangle V(\bar{x}_k, y_k) < 0$. (This is true regardless of the randomness of α_k.) Substituting $\delta_k = \beta_k = 0$ in equations (7.2), we get $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)(\bar{x}_k - y_k)$.

Consider the case where $\bar{x}_k - y_k < 0$. Since $0 \leq (1-\alpha_k) < 1$, we have $\frac{1}{C_1 - N + (\bar{x}_{k+1} - y_{k+1})} + P_1 < \frac{1}{C_1 - N + (\bar{x}_k - y_k)} + P_1$, and $\frac{1}{C_2 - M - (\bar{x}_k - y_k)} + P_2 < \frac{1}{C_2 - M - (\bar{x}_{k+1} - y_{k+1})} + P_2$. Hence, since $\frac{1}{C_1 - N + (\bar{x}_k - y_k)} + P_1 \leq \frac{1}{C_2 - M - (\bar{x}_k - y_k)} + P_2$, we see that $\frac{1}{C_1 - N + (\bar{x}_{k+1} - y_{k+1})} + P_1 \leq \frac{1}{C_2 - M - (\bar{x}_{k+1} - y_{k+1})} + P_2$.

Now, consider the case where $\bar{x}_k - y_k > 0$. We have $\frac{1}{C_1 - N + (\bar{x}_k - y_k)} + P_1 < \frac{1}{C_1 - N} + P_1$, and $\frac{1}{C_2 - M} + P_2 < \frac{1}{C_2 - M - (\bar{x}_k - y_k)} + P_2$. Since $0 \leq (1-\alpha_k) < 1$, and $\frac{1}{C_1 - N} + P_1 \leq \frac{1}{C_2 - M} + P_2$ from equation (7.4), we see that $\frac{1}{C_1 - N + (\bar{x}_{k+1} - y_{k+1})} + P_1 \leq \frac{1}{C_2 - M - (\bar{x}_{k+1} - y_{k+1})} + P_2$.

Similarly, we see that $\frac{M + (\bar{x}_{k+1} - y_{k+1})}{C_2} \leq \frac{N - (\bar{x}_{k+1} - y_{k+1})}{C_1}$. Therefore, $(\bar{x}_{k+1}, y_{k+1}) \epsilon D_1$, and $\delta_{k+1} = \beta_{k+1} = 0$. Consequently, starting at any point in D_1, the trajectory stays inside D_1 approaching the origin (as shown in Figure 7.1). □

From lemma 7.1.1 and the Liapunov stability theory [86], we have the following theorem:

Theorem 7.1.1 *For TOS1, any starting state in D_1 (the shaded area in Figure 7.1) leads to the origin, regardless of the values of α_k.* □

The region D_1 is called *domain of attraction* corresponding to the origin [10, 61] because it constitutes a set of starting states for which the iteration converges to the origin. In D_1, the iteration is said to be a *contraction*, since $V(\bar{x}_{k+1}, y_{k+1}) < V(\bar{x}_k, y_k)$ for all $(\bar{x}_k, y_k) \neq (0,0)$ along the trajectory. It is important to observe that the domain of attraction contains all the system states for which $T_1 \leq T_2$ and $\rho_2 \leq \rho_1$. Also, note that starting at any point in D_1, $\delta_k = \beta_k = 0$ for all $(\bar{x}_k, y_k)$ along the trajectory.

With TOS2, the system behavior is described by the same difference equations (7.2) except that δ_k is defined as

$$\delta_k = \begin{cases} 0 & \text{if } \frac{1}{C_{1,k}^{\mathit{eff}} - N + \bar{x}_k} + P_1 \leq \frac{1}{C_{2,k}^{\mathit{eff}} - \bar{x}_k} + P_2 \\ 1 & \text{otherwise} \end{cases}$$

At the fixed point, $x_k \to N, \bar{x}_k \to 0$, $y_k \to 0, \bar{y}_k \to M$. Then, from equations (6.5), we have $C_{2,k}^{\mathit{eff}} \to C_2 - 0.5M$, and $C_{1,k}^{\mathit{eff}} \to C_1$. For equations (7.2) to be satisfied at the fixed point, $\delta_k \to 0$ and $\beta_k \to 0$. Then necessary conditions for convergence to the origin are:

$$\begin{aligned} \frac{1}{C_1 - N} + P_1 \quad &\leq \quad \frac{1}{C_2 - 0.5M} + P_2 \\ \frac{M}{C_2} \quad &\leq \quad \frac{N}{C_1} \end{aligned} \tag{7.7}$$

As we have done with TOS1, we want to show that $V(\bar{x}, y)$, defined in (7.5), is a Liapunov function in some region around the origin. Call this region D_2. The goal is to show that starting at any point in D_2, $\delta_k = \beta_k = 0$ for all $(\bar{x}_k, y_k)$ along the trajectory.

$\frac{1}{C_{1,k}^{\mathit{eff}} - N + \bar{x}_k} + P_1 \leq \frac{1}{C_{2,k}^{\mathit{eff}} - \bar{x}_k} + P_2$ implies $\delta_k = 0$. $\frac{M + (\bar{x}_k - y_k)}{C_2} \leq \frac{N - (\bar{x}_k - y_k)}{C_1}$ implies $\beta_k = 0$. Because the expressions for $C_{i,k}^{\mathit{eff}}$ are hard to work with, we try to find simpler expressions, say $C_{i,k}^{simp}$, such that $\frac{1}{C_{1,k}^{\mathit{eff}} - N + \bar{x}_k} + P_1 \leq \frac{1}{C_{1,k}^{simp} - N + \bar{x}_k} + P_1$, and $\frac{1}{C_{2,k}^{simp} - \bar{x}_k} + P_2 \leq \frac{1}{C_{2,k}^{\mathit{eff}} - \bar{x}_k} + P_2$ (then $\frac{1}{C_{1,k}^{simp} - N + \bar{x}_k} + P_1 \leq \frac{1}{C_{2,k}^{simp} - \bar{x}_k} + P_2$ implies $\delta_k = 0$). We would then define $D_2 = \{(\bar{x}, y) : \frac{1}{C_1^{simp} - N + \bar{x}} + P_1 \leq \frac{1}{C_2^{simp} - \bar{x}} + P_2 \ \wedge \ \frac{M + (\bar{x} - y)}{C_2} \leq \frac{N - (\bar{x} - y)}{C_1} \ \wedge \ 0 \leq \bar{x} \leq N \ \wedge \ 0 \leq y \leq M\}$, and attempt to show that $V(\bar{x}, y)$, defined in (7.5), is a Liapunov function in D_2.

To do this, we need an upper bound on $C_{2,k}^{\mathit{eff}}$, and a lower bound on $C_{1,k}^{\mathit{eff}}$. From equations (6.5), we see that $C_{2,k}^{\mathit{eff}} \leq C_2 - 0.5(\bar{y}_k - \bar{x}_k)$. We could not find an appropriate lower bound on $C_{1,k}^{\mathit{eff}}$. So we made the approximation $C_{1,k}^{\mathit{eff}} \approx C_1 - 0.5y_k$ (the accuracy of this is discussed below).

We thus have $D_2 = \{(\bar{x}, y) : \frac{1}{(C_1 - 0.5y) - N + \bar{x}} + P_1 \leq \frac{1}{(C_2 - 0.5(\bar{y} - \bar{x})) - \bar{x}} + P_2 \ \wedge \ \frac{M + (\bar{x} - y)}{C_2} \leq \frac{N - (\bar{x} - y)}{C_1} \ \wedge \ 0 \leq \bar{x} \leq N \ \wedge \ 0 \leq y \leq M\}$. Figure 7.2 depicts this region.

Lemma 7.1.2 *Assuming $C_{1,k}^{\mathit{eff}} \approx C_1 - 0.5y_k$, $V(\bar{x}, y)$ is a Liapunov function in D_2.* □

The proof of the above lemma is similar to the proof of Lemma 7.1.1, and is given in Appendix C. From lemma 7.1.2 and the Liapunov stability theory [86], we have the following theorem:

Theorem 7.1.2 *For TOS2, assuming $C_{1,k}^{\mathit{eff}} \approx C_1 - 0.5y_k$, any starting state in D_2 (the shaded area in Figure 7.2) leads to the origin, regardless of the values of α_k.* □

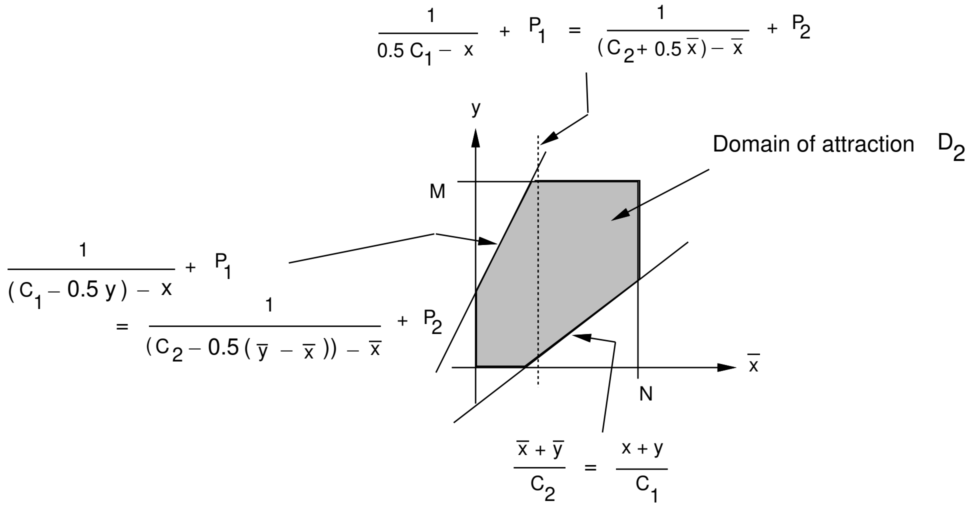

Figure 7.2: Domain of attraction for TOS2.

7.1.1 Effect of α_k on system behavior

The domains of attraction we have just found (cf. Theorems 7.1.1 and 7.1.2) are not the largest, i.e. there may be points outside the domains which lead to the origin. This depends on the values of α_k. In particular, the domains are indeed largest for high enough values of α_k. On the other hand, they are not for small values of α_k. The following theorem shows this for TOS1. The proof is given in Appendix C.

Theorem 7.1.3 *For TOS1, starting at any point in area A or area B (Figure 7.1), the following hold:*

(i) If $\alpha_{MIN} > max(\frac{N-L_1}{M+N}, \frac{M+L_2}{M+N})$ then the iteration does not converge to the origin and locks into a limit cycle oscillating between states in areas A and B.

(ii) If $\alpha_{MAX} \leq min(\frac{L_2}{N}, \frac{-L_1}{M})$ then the iteration converges to the origin.

where

$$\begin{aligned}
L_1 &= \frac{H\ S\ - 2 + \sqrt{(H\ S\ - 2)^2 + 4\ H[H\ (C_1 - N)\ (C_2 - M) + S]}}{2H} \\
H &= P_1 - P_2 \\
S &= C_2 - M - C_1 + N \\
L_2 &= \frac{C_2 N - C_1 M}{C_1 + C_2}
\end{aligned}$$

□

Theorem 7.1.3 indicates that for high enough values of α_k, the system may not converge to isolation, and rather oscillates with both traffic classes shifting *simultaneously* at each routing update. Such simultaneous traffic shifts result in a bad performance, i.e. higher delay and network under-utilization. This effect increases as α_k increases. (In practice, α_k depend on several factors, and can be quite high [36].)

Consider the simple case where $\alpha_k = 1$, for all k. It can be seen that isolation, whenever possible[2], provides the optimal performance for both traffic classes [30]. In this case, we are constrained to use a single path for each traffic class. Thus, in order to maximize the throughput of the throughput-sensitive traffic, we should send its packets over the maximum capacity link, i.e. path 2. Then, in order to minimize the packet delay of the delay-sensitive traffic, we should send its packets over the minimum packet delay link, i.e. path 1. Note that routing the delay-sensitive traffic (also) on path 2 would result in a higher delay compared to the delay of (the unused) path 1.

Note that the above result agrees with our argument about the benefits of isolation, which was made in Section 6.2.2.

Referring to Figures 7.1 and 7.2, we can conclude that for high enough α_k, TOS2 has a larger domain of attraction corresponding to isolation than TOS1. This conclusion is not affected by our approximation $C^{eff}_{1,k} \approx C_1 - 0.5y_k$, which was made in Theorem 7.1.2 for TOS2. In fact, it can be shown that $C^{eff}_{1,k} \leq C_1 - 0.5y_k$. Regardless of that, we found that our approximation is only slightly optimistic. In particular, our monte-carlo simulations [75, 73] show that starting at any point in D_2 satisfying $\frac{1}{0.5C_1-N+\bar{x}} + P_1 \leq \frac{1}{(C_2+0.5\bar{x})-\bar{x}} + P_2$, the iteration indeed leads to the origin. Those points also satisfy $\frac{1}{(C_1-0.5y)-N+\bar{x}} + P_1 \leq \frac{1}{(C_2-0.5(\bar{y}-\bar{x}))-\bar{x}} + P_2$. This is because $\frac{1}{(C_1-0.5y)-N+\bar{x}} + P_1 \leq \frac{1}{0.5C_1-N+\bar{x}} + P_1$ (with equality occurring when $y = C_1$), and $\frac{1}{(C_2+0.5\bar{x})-\bar{x}} + P_2 \leq \frac{1}{(C_2-0.5(\bar{y}-\bar{x}))-\bar{x}} + P_2$. Some points in D_2 not satisfying $\frac{1}{0.5C_1-N+\bar{x}} + P_1 \leq \frac{1}{(C_2+0.5\bar{x})-\bar{x}} + P_2$ may not, however, lead to the origin. Such points constitute a small part of D_2, and hence the approximation does not affect our conclusion that TOS2 has a larger domain of attraction corresponding to isolation. In particular, isolation occurs for higher values of y_0 with TOS2 than with TOS1.

Figures 7.3 and 7.4 show the domains of attraction corresponding to isolation for TOS1 and TOS2, respectively, obtained by monte-carlo simulations for $C_1 = 30, C_2 = 50, P_1 = 0.5, P_2 = 1, N = 20, M = 30$, and $\alpha_k \,\epsilon\, [0.8, 1]$.

Theorem 7.1.3 also indicates that for small enough values of α_k, the system reaches isolation for all starting states and for all system configurations satisfying the necessary conditions for isolation. Figure 7.5 depicts the *load region*, i.e. values of (N, M), defined by $\{(N, M) : \frac{1}{C_1-N} + P_1 \leq \frac{1}{C_2} + P_2 \;\wedge\; \frac{M}{C_2} \leq \frac{N}{C_1}\}$. In this region, the necessary conditions for isolation, defined in (7.4) and (7.7), for both TOS1 and TOS2 are satisfied. This region thus defines sufficient conditions to reach isolation for low enough α_k for both TOS1 and TOS2. This is in agreement with our monte-carlo simulations obtained in [75, 73] which also show that in this region, TOS2 gives much better

[2]Which means that the necessary conditions to reach isolation are satisfied, i.e. the delay of path 1 carrying all delay-sensitive traffic is less than or equal the delay of path 2 carrying all throughput-sensitive traffic, with path 2's utilization being less than or equal path 1's utilization.

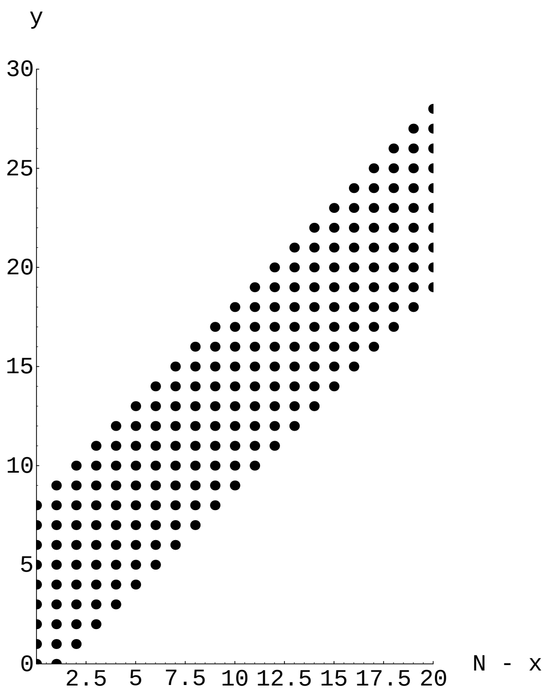

Figure 7.3: Numerical example: Domain of attraction for TOS1.

transient performance than TOS1, i.e. less oscillations and much faster convergence to isolation (future work remains to obtain transient measures analytically).

For $C_1 = 30, C_2 = 50, P_1 = 0.5, P_2 = 1$, and for low to moderate values of α_k ($\alpha_{MAX} \leq 0.6$), Figure 7.6 shows the values of N and M leading to isolation starting from any state (we consider only values of N and M such that $N + M < C_1 + C_2$). Figures 7.7 and 7.8 show the average delays for TOS1 and TOS2, respectively, for $N = 20, M = 25, x_0 = 5, y_0 = 10$ and $\alpha_k \,\epsilon\, [0.4, 0.6]$.

7.1.2 Sufficient conditions independent of the starting state

In Theorems 7.1.1 and 7.1.2, we derived sufficient conditions for the system to reach isolation as a function of the starting state. Now, using these theorems, we derive sufficient conditions for isolation independent of the starting state and the values of α_k.[3] Figure 7.9 depicts the load regions defined by these sufficient conditions.

Theorem 7.1.4

(i) For TOS1, the conditions $N + M \leq C_1 - \frac{1}{(\frac{1}{C_2} + P_2 - P_1)}$ *and* $\frac{M}{C_2} \leq \frac{N}{C_1}$ *(regions B and C in Figure 7.9) are sufficient for isolation (regardless of the starting state and the values of* α_k*).*

[3] These conditions were obtained in [75, 73] in a different way using a worst-case analysis.

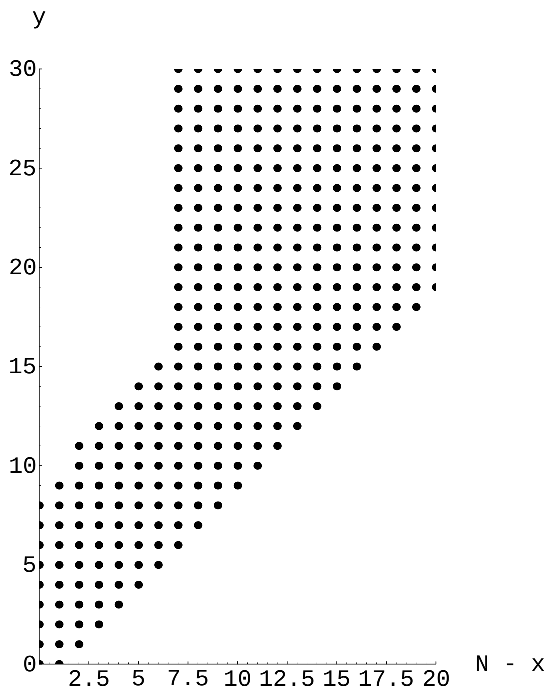

Figure 7.4: Numerical example: Domain of attraction for TOS2.

(ii) For TOS2, the corresponding conditions are $N \leq 0.5C_1 - \frac{1}{(\frac{1}{C_2}+P_2-P_1)}$ *and* $\frac{M}{C_2} \leq \frac{N}{C_1}$ *(regions A and C in Figure 7.9).* □

The proof of Theorem 7.1.4 is given in Appendix C. Referring to Figure 7.9, we observe that for N less than $0.5C_1 - \frac{1}{(\frac{1}{C_2}+P_2-P_1)}$, TOS2 provides isolation (regardless of the starting state and the values of α_k) for higher values of M than TOS1. This effect becomes more pronounced as C_2 increases.

Note that the conditions of Theorem 7.1.4 capture less information than the ones given by Theorems 7.1.1 and 7.1.2; i.e. there can be a system configuration that does not satisfy the sufficient conditions of Theorem 7.1.4, but still reaches isolation for some starting states or values of α_k.

7.2 Comments and Extensions

We found our proposed scheme, TOS2, very effective in providing lower end-to-end delays in a typical situation where the proportion of delay-sensitive traffic is small compared to the throughput-sensitive traffic. This is because our scheme, by using structured link queueing, reduces queueing delays for the delay-sensitive traffic, hence improving the performance of delay-based routing and providing more stable routes for this traffic. At the same time, the link utilization metric isolates the throughput-sensitive traffic which takes the under-utilized routes, resulting in a better overall

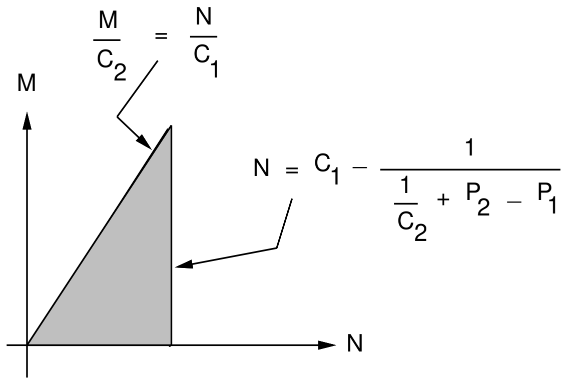

Figure 7.5: Load region for isolation for low enough α_k for both TOS1 and TOS2.

network performance. In general, we have shown how in an integrated services environment, routing with some form of non-FCFS scheduling support can provide significant performance improvement.

We assumed unbounded buffer space at the links. In practice, the buffer space would have to be allocated to the different TOS classes. We would expect the same conclusions to hold. We also assumed a datagram (best-effort) service model of the sort currently used in the Internet. Extensive effort is currently underway to extend this Internet service model to support other services, including guaranteed service [16, 22, 93]. In such an extended service environment, our findings would still apply to traffic classes requiring best-effort service. In addition, our approach appears scalable as each router needs to maintain information only about each TOS class rather than each connection. This also makes the implementation of the non-FCFS queueing discipline in the routers simple and inexpensive.

It is obvious that TOS2 relies upon the delay-sensitive traffic being a small proportion of the overall traffic. When the proportion of delay-sensitive traffic is not small, TOS2 performs worse than TOS1 because the delay-sensitive traffic in TOS2 suffers from higher queueing delay than in TOS1 [72]. However, the performance of TOS2 can be restored by simply allowing, for example, the delay-sensitive queue to get two turns for every turn of the throughput-sensitive queue. The natural extension of this would be to give delay-sensitive packets absolute priority over throughput-sensitive packets. However, giving delay-sensitive packets absolute priority may not be desirable because throughput-sensitive packets may not get transmitted (i.e. blocked) causing queues of this traffic class to build up, and consequently providing low throughput for the traffic class requesting high throughput service. Future work remains to similarly analyze such situations and propose a scheduling discipline that dynamically allocates turns to the TOS queues based on traffic estimates.

There might be other possible approaches to adaptive next-hop TOS routing. One approach [109] is to maintain for delay-sensitive traffic two minimum propagation delay routes and use the secondary route when the primary route becomes congested. This approach attempts to avoid the bad effect that queueing delay may have when it dominates measured link delay in delay-based routing.

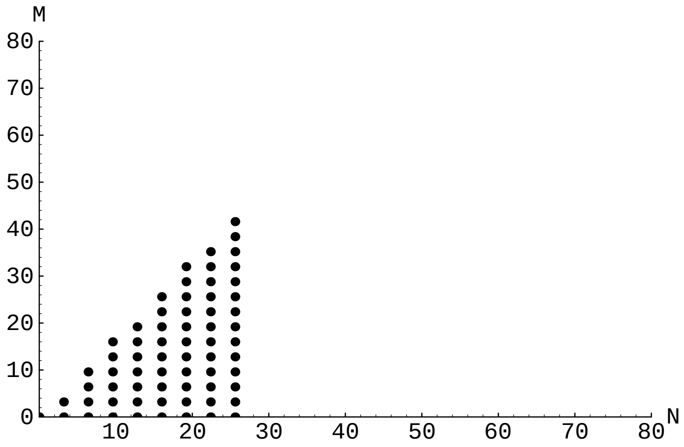

Figure 7.6: Numerical example: Load region for isolation at low to moderate α_k for both TOS1 and TOS2.

However, it is not clear how to split the delay-sensitive traffic between the two available routes without causing severe oscillations while at the same time reducing queueing delays to provide a low delay service. Further research is needed to explore this area.

It is interesting to observe the relationship between our work here and work in the virtual-circuit literature. We took the concept of isolating link resources, often used in virtual-circuit models, and applied it to a datagram model. This induced isolation of routes for the TOS classes, resulting in an improved overall network performance. It would be interesting to see if the concept of route isolation is also applicable in virtual-circuit models. A related idea in virtual-circuit models is the packing of low-bandwidth connections over a subset of the routes leaving the remaining routes free for high-bandwidth connections, thus in effect isolating both types of connections [45].

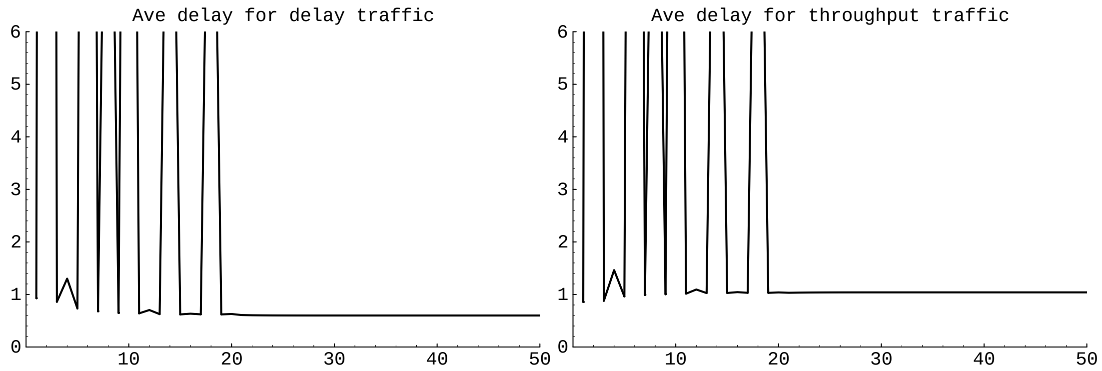

Figure 7.7: Numerical example: Average delays versus time k for TOS1 ($\alpha_k \in [0.4, 0.6]$).

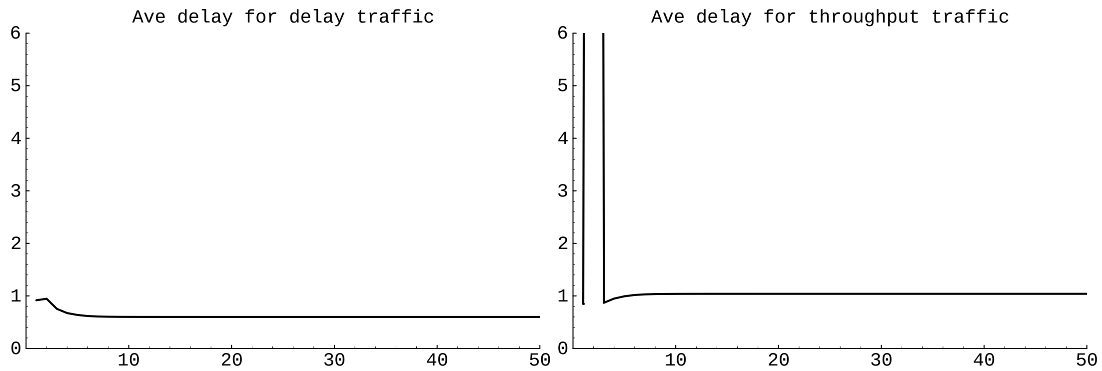

Figure 7.8: Numerical example: Average delays versus time k for TOS2 ($\alpha_k \in [0.4, 0.6]$).

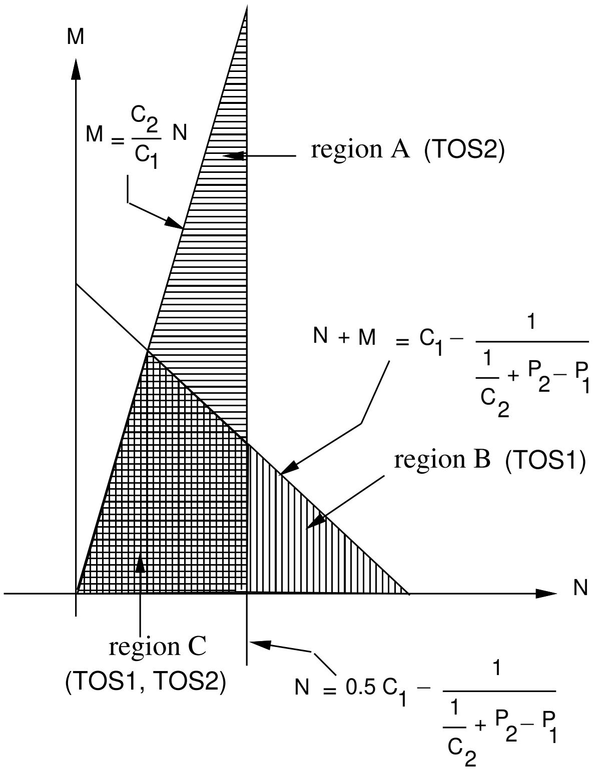

Figure 7.9: Load regions satisfying sufficient conditions for isolation independent of starting state.

Chapter 8

Conclusions and Future Research Directions

Integrated services networks have often been analyzed under steady-state conditions. In this dissertation, we presented a numerical-analytical method, the Z-iteration, to rapidly evaluate detailed and dynamic models of integrated services networks as well as other MCMR systems. Our results indicate that the method gives approximate, yet accurate, instantaneous performance measures and provides significant computational savings over discrete-event simulation. We applied our method to compare different adaptive routing algorithms.

There are several areas for future work. One area is to examine routing schemes that distinguish between different types of traffic (e.g., low-throughput voice and high-throughput video), computing a different set of routes for each type. For example, for a particular traffic type with very stringent QoS requirements, we could restrict the set of candidate paths to only minimum-hop paths, while for other traffic types the set could also include next-to-minimum-hop paths. One would examine the capability of the routing scheme to distribute connections of each type in a way that increases the network throughput, and also the responsiveness of the routing scheme to failures and repairs.

Another area is to examine admission controls, other than the complete-sharing policy, that block some connection setup requests even if their admission is feasible, possibly in order to reduce the chance of future blocking of connections of other types. In this case, blocking would occur at more feasible states [104, 50].

Another area is to investigate policies other than the equal allocation policy for dividing the end-to-end QoS requirement among the links of a route. These policies would take into account the current link loads as measured in the last routing update period. Other QoS requirements such as packet loss can be considered.

We plan to extend the numerical-analytical techniques embodied in the Z-iteration. Such techniques are much needed in system design phase where a good compromise between accuracy and speed is crucial. We intend to build evaluation tools for integrated networks and other complex adaptive systems such as distributed operating systems.

In the last part of the dissertation, we presented a quasi-static model to analyze the behavior of TOS schemes in datagram delivery systems. We obtained stability conditions using the Liapunov function method. We showed how in an integrated services environment, routing with some form of non-FCFS scheduling support can provide significant performance improvement. We intend to extend our analysis to obtain transient characteristics such as convergence time. Our analysis demonstrates the interaction between adaptive routing and link scheduling. Future work is also needed to explore the interaction between all components of the congestion control problem, namely, scheduling, flow control, and routing, on more detailed network models with arbitrary topologies.

We intend to integrate quasi-static evaluation techniques into our Z-iteration based tools whenever appropriate. This would further reduce computation times while retaining accuracy.

We hope to use our tools to investigate issues in integrated networks that are not yet well understood. We explain some of these issues below.

Internet-style versus virtual-path routing: There are two styles of virtual-circuit (VC) routing. In Internet-style routing, VCs are routed over paths in the physical network topology. Admission control is done on every physical link along the path provided by the routing component. In virtual-path based routing, a virtual-path (VP) is typically specified between every two end-systems and set up over a sequence of physical links. Enough bandwidth is statically allocated to each VP so that queueing occurs only at the first physical link of the VP. Several VPs can share the same physical link. Thus, the physical network is transformed to a (logically) fully-connected network of non-interacting VPs. VCs are then routed over one-hop and two-hop VPs, and thus admission control is done on at most two physical links.

Obviously, admission control is much faster with VP based routing than with Internet-style routing. However, the network may be under-utilized because of the potential loss of statistical multiplexing due to the static allocation of resources to VPs [49]. This has led to attempts to dynamically allocate resources to VPs [88]. However, depending on the rate at which the VP resource allocations are varied compared to traffic demands, this approach may not be effective or may give rise to massive instability because of the strong interaction between VP resource allocation and routing/admission control.

It is important to examine both routing styles, evaluating the tradeoffs between admission control delay and network utilization. Large integrated networks (or internetworks) with non-trivial connectivity introduce scalability issues to the problem. Scalable solutions exist with both styles of routing (e.g. [11, 4, 2, 102, 64]). Future work is needed to examine such solutions and determine conditions under which each solution is effective.

Multicast algorithms: Another issue to investigate is multicast algorithms, where the interaction among the control components is much stronger than in unicast. Many algorithms have been proposed to construct multicast paths, e.g. core-based algorithms [9]. It is not clear, however, how they compare under conditions of dynamic workload and topology. Comparisons have often been made only in terms of worst-case bounds or under static conditions. Clearly, we also do not know how these various algorithms would compare when resource reservations are performed along the multicast path to provide guaranteed services. What policies should we use to allocate resources on the different links, etc.? For example, how do these algorithms interact with reservation protocols like RSVP [82].

Integration of wireless and wired networks: Another issue to investigate is the integration of wireless and wired networks. In particular, it is interesting to investigate ways to extend qualities of service over wireless segments. This is a challenging and open problem. Wireless networks have unique characteristics related to loss, bandwidth, mobility, etc., which adds new dimensions to the problem of QoS support [56, 66].

Appendix A

Simulation Details for Type-of-Service Datagram Network

In this appendix, we describe the simulation parameters, and the performance measures. We also give details of simulated scenarios and plots.

Parameters

All links have bandwidths of 1.5 Mbit/sec for the low-speed version, and 100 Mbit/sec for the high-speed version. Each node's outgoing link costs are updated regularly, with inter-update time uniformly distributed with mean 10 seconds [78] and standard deviation 1 second.[1] The factor b, used in the link cost calculation, is 0.8.[2] A data source is represented as a Markov chain with two states: a busy state and an idle state. In the busy state, the source produces a (geometrically distributed) number of data packets with some constant inter-packet generation time. The source then stays idle for an exponentially distributed duration before starting the transmission of the next train (burst) of packets. (This traffic model has been used in many studies, e.g. [107].) Unless otherwise indicated, all sources have the following parameters: for the low-speed case, the data packet length equals 128 bytes, the inter–packet generation time is 150 msec, the average train size is 100 packets, and the average idle duration is 2 seconds (this corresponds to an average packet rate of about 0.006 packet/msec.); for the high-speed case, the data packet length is 5000 bytes, the inter–packet generation time is 50 msec, the average train size is 1000 packets, and the average idle duration is 2 seconds (this corresponds to an average packet rate of about 0.02 packet/msec.).

[1] Update intervals at different nodes are independent.

[2] We chose the factor 0.8 after experimenting with other values. A small value such as 0.5 makes the routing algorithm adapt slowly. Whereas a high value such as 1.0 may result in an unstable behavior.

Performance Measures

We consider average measures of throughput, delay and load. An average measure is based on statistics collected over a large measurement interval, which is the duration of the simulation except for an initial "startup interval" (to eliminate transient effects due to empty initial network). Thus:

- *Throughput.* Total number of data bytes received at destinations during the measurement interval divided by the length of the measurement interval.
- *Delay.* Total delay of all data packets received at destinations during the measurement interval divided by the number of those data packets, where *delay of a data packet* is defined to be the time difference between sending a packet and receiving it at the corresponding destination.
- *Data Load.* Fraction of the network capacity, i.e. sum of all link capacities, used by data packets during the measurement interval.
- *Throughput(T).* Total number of data bytes received at destinations for the throughput-sensitive traffic during the measurement interval divided by the length of the measurement interval.
- *Delay(D).* Total delay of all delay-sensitive data packets received at destinations during the measurement interval divided by the number of those data packets.

Results

Here, we present details of scenarios simulated along with plots of the observed (steady–state) performance measures, namely *throughput*, *delay*, *throughput*(T), *delay*(D), and *data load*. In our simulations, 95% confidence intervals were computed using the method of independent replications [67, 97]. In particular, a measure, say x, is obtained as $\frac{x_1+x_2+\cdots+x_n}{n}$, where the $x_1, x_2, \ldots, x_n$ are the measures obtained using the different independent simulation runs. In all cases, the size of the confidence intervals is less than 2% of the mean.

We now present our simulation results. Although we show results only for uniform workload, we obtained similar results for the skewed workload we investigated. We also obtained similar results for workload that has different parameters for each traffic type. In case (C) below, we consider a smaller packet size for the delay-sensitive sources, namely 64 bytes rather than 128 bytes. We have scaled the delay plots for clarity, so delay values higher than 100 milliseconds are not shown.

(A) Low-speed, varying U(T), fixed U(D), equal packet sizes.

Figure A.1 shows $delay(D)$ and *delay* versus $U(T)$ in the range 20 to 26, for a fixed $U(D) = 8$. The delay-sensitive traffic constitutes almost 25% of the total traffic. Ignoring differences at low values of $U(T)$, TOS1 performs the worst, becoming saturated around $U(T) = 22$ (corresponding to almost 60% data load). Interestingly, UTIL which does *not* use any TOS facility performs better than TOS1. TOS2 performs the best. At $U(T) = 24$, UTIL's *delay* is 53% higher than TOS2's, and TOS1's *delay* is 1322% higher than TOS2's. Note that UTIL performs worse than TOS2 over the whole range of $U(T)$. At low values of $U(T)$, UTIL performs the worst. At $U(T) = 20$, UTIL has about 32% higher *delay* than both TOS2 and TOS1.

Figure A.2 shows *data load*, $throughput(T)$, and *throughput* versus $U(T)$. Observe that the data load for TOS1 increases as TOS1 becomes saturated. This indicates the use of longer routes, and consequently higher delay. Both $throughput(T)$ and *throughput* increase linearly with the workload, and they are the same for all schemes. This shows that the system is stable for all schemes.

(B) High-speed, varying U(T), fixed U(D), equal packet sizes.

Figure A.3 shows $delay(D)$ and *delay* versus $U(T)$ in the range 8 to 18, for a fixed $U(D) = 4$. The delay-sensitive traffic constitutes almost 25% of the total traffic. As observed in the low-speed case (A), ignoring differences at low values of $U(T)$, TOS2 performs the best, followed by UTIL, and then TOS1. TOS1 reaches saturation sooner, around $U(T) = 16$ (corresponding to a data load of almost 61%). TOS1 has about 758% higher *delay* than TOS2 at $U(T) = 18$. Note that for the same data load, the difference in delay is less significant than in case (A). This is due to the fact that in a high-speed network, queueing delay is less significant due to small transmission times. For example, in the high-speed network, transmission time of the 5000-byte data packet on the 100Mbit/s link is 0.0004 sec. Whereas in the low-speed network, transmission time of the 128-byte data packet on the 1.5Mbit/s link is 0.7 sec. This fact reduces the effect of bad oscillations inherent in delay-based routing when queueing delays are significant.

Figure A.4 shows *data load*, $throughput(T)$ and *throughput* versus $U(T)$. Again, as in case (A), $throughput(T)$ and *throughput* increase linearly with the workload for all schemes. Henceforth, we do not show plots for throughput. We also do not show plots for UTIL since the utilization-based metric, as we have observed, does not necessarily result in minimum delay routes.[3]

[3] We have also studied a non-TOS scheme similar to UTIL except that it uses the delay metric rather than the utilization metric (i.e. $LinkCost(D)$ is used for all traffic); however we do not show plots for it since it performed badly at heavy load, as expected.

(C) Low-speed, varying U(T), fixed U(D), unequal packet sizes.

Figure A.5 shows *delay*(D) and *delay* versus $U(T)$ in the range 16 to 24, for a fixed $U(D) = 16$. The delay-sensitive traffic constitutes almost 28% of the total traffic. As observed in (A), TOS2 performs better than TOS1. TOS1 reaches saturation sooner, around $U(T) = 18$ (corresponding to a data load of almost 55%). TOS1 has about 1800% higher *delay* than TOS2 at $U(T) = 20$.

Figure A.6 shows *data load* versus $U(T)$. Observe that TOS1 reaches saturation at a data load which is smaller than in (A). This is because here delay-sensitive traffic has smaller packet sizes, thus suffering higher delays with TOS1.

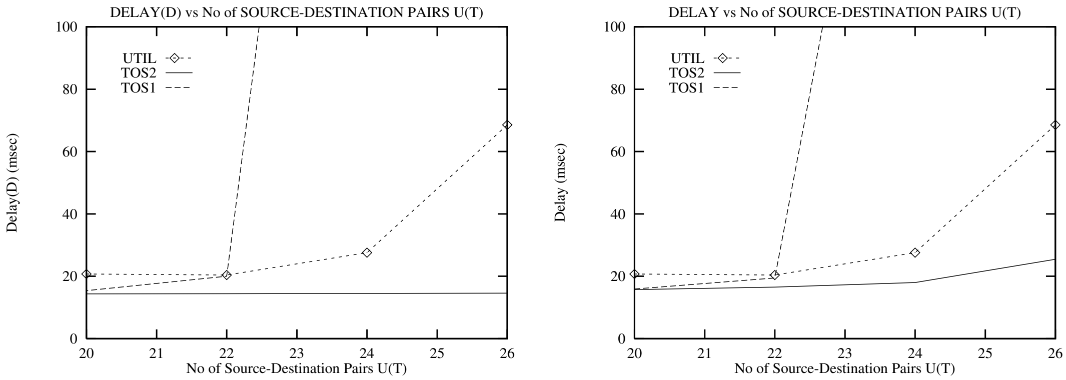

Figure A.1: Low-speed. Equal packet sizes. *Delay*(D) and *delay* vs $U(T)$ for $U(D) = 8$.

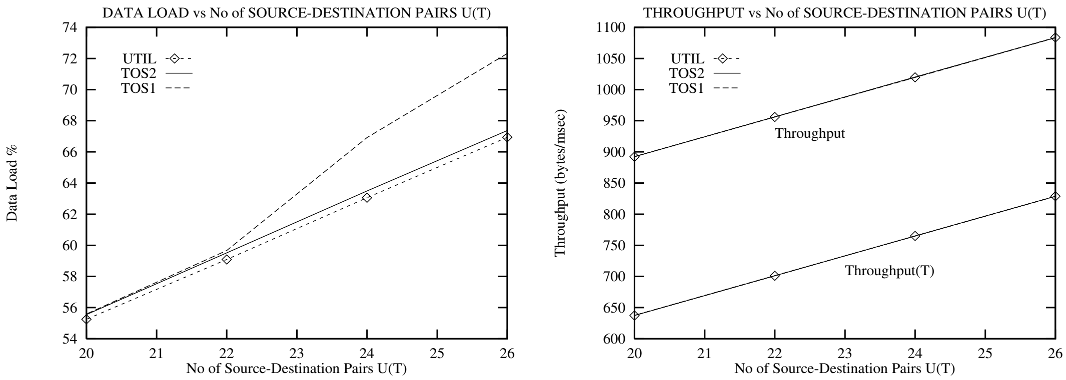

Figure A.2: Low-speed. Equal packet sizes. *Data load* and *throughput* vs $U(T)$ for $U(D) = 8$.

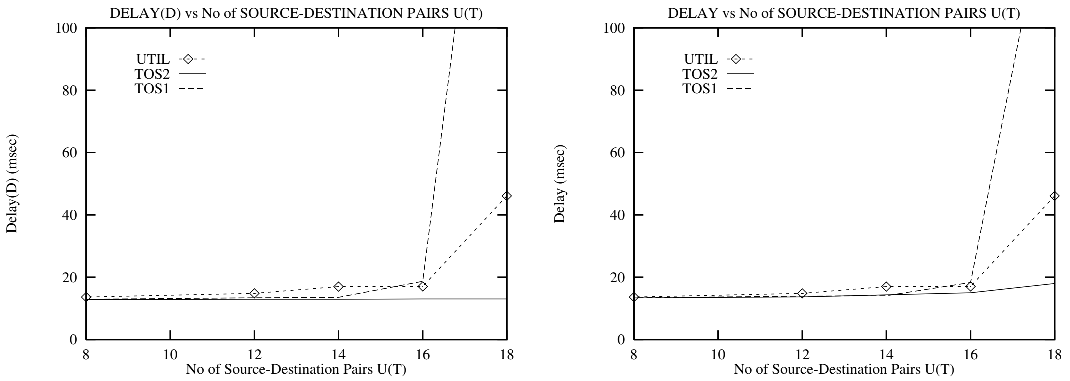

Figure A.3: High-speed. Equal packet sizes. $Delay(D)$ and $delay$ vs $U(T)$ for $U(D) = 4$.

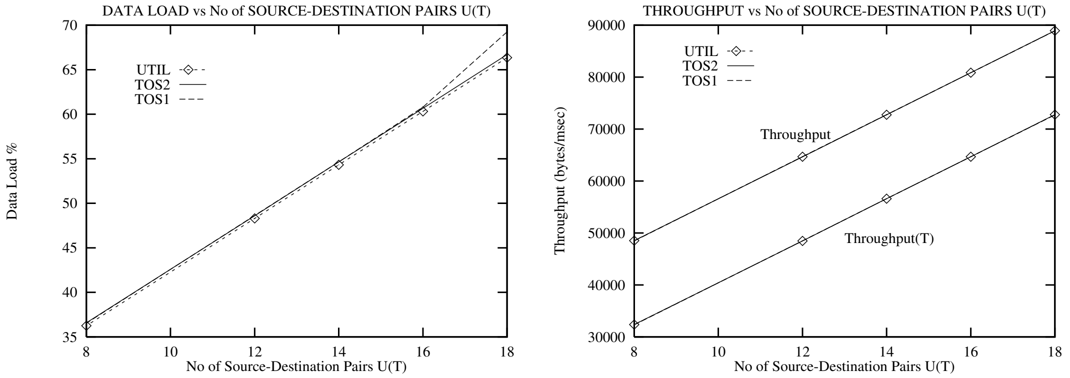

Figure A.4: High-speed. Equal packet sizes. $Data\ load$ and $throughput$ vs $U(T)$ for $U(D) = 4$.

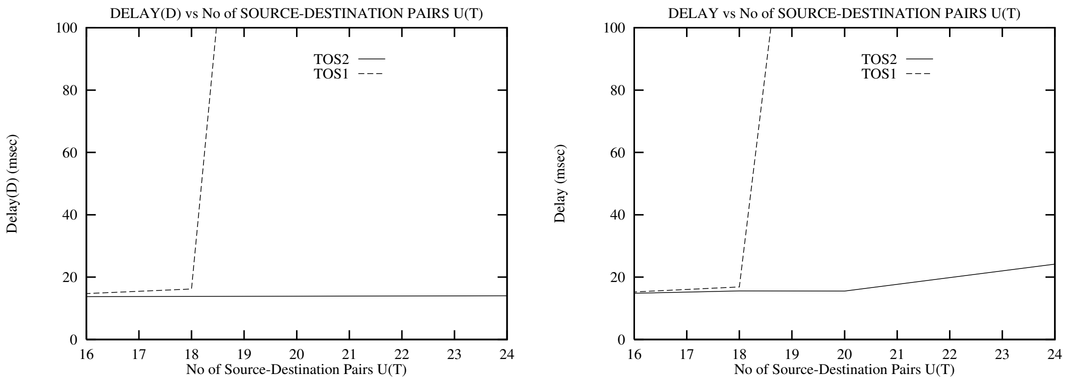

Figure A.5: Low-speed. Unequal packet sizes. $Delay(D)$ and $delay$ vs $U(T)$ for $U(D) = 16$.

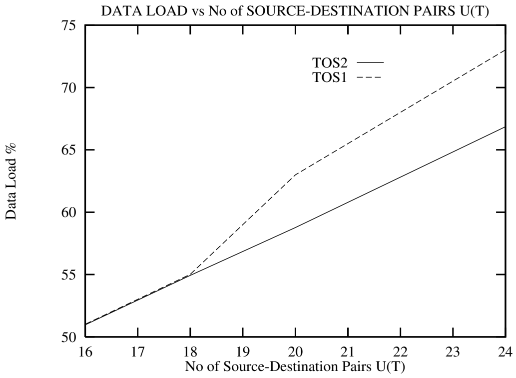

Figure A.6: Low-speed. Unequal packet sizes. *Data load* vs $U(T)$ for $U(D) = 16$.

Appendix B

Computation of Effective Capacities for Quasi-Static Model

Let's consider two queues Q_X and Q_Y with arrival rates X and Y, respectively, *equally* sharing a link with capacity C. Define C_X^{eff} and C_Y^{eff} as the effective link capacity available for traffic X and Y, respectively. Also, let n_X and n_Y be the number of packets in Q_X and Q_Y, respectively. Thus,

$$\begin{aligned} C_X^{eff} &= C \times Prob[n_Y = 0] + 0.5C \times Prob[n_Y > 0] \\ C_Y^{eff} &= C \times Prob[n_X = 0] + 0.5C \times Prob[n_X > 0] \end{aligned}$$

Note that $C_X^{eff} \geq 0.5C$ ($C_Y^{eff} \geq 0.5C$), with the worst-case occurring when $Q_Y(Q_X)$ is *always* not empty.

Assuming each queue is $M/M/1$, and substituting

$$\begin{aligned} Prob[n_X = 0] &= 1 - Prob[n_X > 0] = 1 - X/C_X^{eff}, \\ Prob[n_Y = 0] &= 1 - Prob[n_Y > 0] = 1 - Y/C_Y^{eff} \end{aligned}$$

we obtain two equations in the two unknowns C_X^{eff} and C_Y^{eff}. Solving them, we get

$$\begin{aligned} C_X^{eff} &= \frac{(C - 0.5(Y-X)) + \sqrt{(C-0.5(Y-X))^2 - 2CX}}{2} \\ C_Y^{eff} &= C_X^{eff} + 0.5(Y - X) \end{aligned}$$

Appendix C

Proofs for Quasi-Static Model

Proof of Lemma 7.1.2

Consider a point $(\bar{x}_k, y_k)\ \epsilon\ D_2 - \{(0,0)\}$. $\frac{1}{(C_1-0.5y_k)-N+\bar{x}_k} + P_1 \leq \frac{1}{(C_2-0.5(\bar{y}_k-\bar{x}_k))-\bar{x}_k} + P_2$ implies $\frac{1}{C^{eff}_{1,k}-N+\bar{x}_k} + P_1 \leq \frac{1}{C^{eff}_{2,k}-\bar{x}_k} + P_2$, and hence $\delta_k = 0$. Let's rewrite $\frac{1}{(C_1-0.5y_k)-N+\bar{x}_k} + P_1 \leq \frac{1}{(C_2-0.5(\bar{y}_k-\bar{x}_k))-\bar{x}_k} + P_2$ as $\frac{1}{C_1-N+(\bar{x}_k-0.5y_k)} + P_1 \leq \frac{1}{C_2-0.5M-0.5(\bar{x}_k-y_k)} + P_2$. Since $\delta_k = \beta_k = 0$ then $\bar{x}_{k+1} - 0.5y_{k+1} = (1-\alpha_k)(\bar{x}_k - 0.5y_k)$, and $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)(\bar{x}_k - y_k)$.

Consider the case where $\bar{x}_k - 0.5y_k < 0$. This implies that $\bar{x}_k - y_k < 0$. Since $0 \leq (1 - \alpha_k) < 1$, we have $\frac{1}{C_1-N+(\bar{x}_{k+1}-0.5y_{k+1})} + P_1 < \frac{1}{C_1-N+(\bar{x}_k-0.5y_k)} + P_1$, and $\frac{1}{C_2-0.5M-0.5(\bar{x}_k-y_k)} + P_2 < \frac{1}{C_2-0.5M-0.5(\bar{x}_{k+1}-y_{k+1})} + P_2$. Hence, since $\frac{1}{C_1-N+(\bar{x}_k-0.5y_k)} + P_1 \leq \frac{1}{C_2-0.5M-0.5(\bar{x}_k-y_k)} + P_2$, we see that $\frac{1}{C_1-N+(\bar{x}_{k+1}-0.5y_{k+1})} + P_1 \leq \frac{1}{C_2-0.5M-0.5(\bar{x}_{k+1}-y_{k+1})} + P_2$.

Now, consider the case where $\bar{x}_k - 0.5y_k > 0$. In this case, either $\bar{x}_k - y_k > 0$ or $\bar{x}_k - y_k < 0$. First, consider the case where $\bar{x}_k - 0.5y_k > 0$ and $\bar{x}_k - y_k > 0$. We have $\frac{1}{C_1-N+(\bar{x}_k-0.5y_k)} + P_1 < \frac{1}{C_1-N} + P_1$, and $\frac{1}{C_2-0.5M} + P_2 < \frac{1}{C_2-0.5M-0.5(\bar{x}_k-y_k)} + P_2$. Since $0 \leq (1-\alpha_k) < 1$, and $\frac{1}{C_1-N} + P_1 \leq \frac{1}{C_2-0.5M} + P_2$ from equation (7.7), we see that $\frac{1}{C_1-N+(\bar{x}_{k+1}-0.5y_{k+1})} + P_1 \leq \frac{1}{C_2-0.5M-0.5(\bar{x}_{k+1}-y_{k+1})} + P_2$. We also see that the latter inequality also holds for the case where $\bar{x}_k - 0.5y_k > 0$ and $\bar{x}_k - y_k < 0$.

Therefore, since in all cases, $\frac{1}{(C_1-0.5y_{k+1})-N+\bar{x}_{k+1}} + P_1 \leq \frac{1}{(C_2-0.5(\bar{y}_{k+1}-\bar{x}_{k+1}))-\bar{x}_{k+1}} + P_2$ then $\frac{1}{C^{eff}_{1,k+1}-N+\bar{x}_{k+1}} + P_1 \leq \frac{1}{C^{eff}_{2,k+1}-\bar{x}_{k+1}} + P_2$. Similarly, we see that $\frac{M+(\bar{x}_{k+1}-y_{k+1})}{C_2} \leq \frac{N-(\bar{x}_{k+1}-y_{k+1})}{C_1}$. Therefore, $(\bar{x}_{k+1}, y_{k+1})\ \epsilon\ D_1$, and $\delta_{k+1} = \beta_{k+1} = 0$. Thus, we see that the iteration is a contraction in D_2.

Proof of Theorem 7.1.3

First, we note that $\frac{M+(\bar{x}_k-y_k)}{C_2} \leq \frac{N-(\bar{x}_k-y_k)}{C_1}$ iff $\bar{x}_k - y_k \leq L_2$. Also, $\frac{1}{C_1-N+(\bar{x}_k-y_k)} + P_1 \leq \frac{1}{C_2-M-(\bar{x}_k-y_k)} + P_2$ iff $\bar{x}_k - y_k \geq L_1$. $L_1 \leq 0$ and $L_2 \geq 0$ are necessary for the domain of attraction D_1 to surround the fixed point $(0,0)$ and thus convergence to be possible.[1]

If $\bar{x}_k - y_k < L_1$ (i.e. $\delta_k = 1, \beta_k = 0$) then $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)\ (\bar{x}_k - y_k) + \alpha_k\ N \geq -(1-\alpha_{MIN})\ M + \alpha_{MIN}\ N$. If $\bar{x}_k - y_k > L_2$ (i.e. $\delta_k = 0, \beta_k = 1$) then $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)\ (\bar{x}_k - y_k) - \alpha_k\ M \leq (1-\alpha_{MIN})\ N - \alpha_{MIN}\ M$. Thus, the following conditions cause TOS1 to lock into a limit cycle for any $(\bar{x}_0, y_0)$ not in the domain of attraction D_1:

$$\begin{aligned} -(1-\alpha_{MIN})\ M + \alpha_{MIN}\ N &> L_2 \\ (1-\alpha_{MIN})\ N - \alpha_{MIN}\ M &< L_1 \end{aligned}$$

This implies $\alpha_{MIN} > max(\frac{N-L_1}{M+N}, \frac{M+L_2}{M+N})$, and part (i) is proved.

If $\bar{x}_k - y_k < L_1$ then $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)\ (\bar{x}_k - y_k) + \alpha_k\ N \leq \alpha_k\ N \leq \alpha_{MAX}\ N$. We also have $\bar{x}_{k+1} - y_{k+1} > \bar{x}_k - y_k$.

If $\bar{x}_k - y_k > L_2$ then $\bar{x}_{k+1} - y_{k+1} = (1-\alpha_k)\ (\bar{x}_k - y_k) - \alpha_k\ M \geq -\alpha_k\ M \geq -\alpha_{MAX}\ M$. We also have $\bar{x}_{k+1} - y_{k+1} < \bar{x}_k - y_k$.

Thus, the following conditions force the iteration to eventually enter the domain of attraction D_1 and converge to the origin for any $(\bar{x}_0, y_0)$:

$$\begin{aligned} \alpha_{MAX}\ N &\leq L_2 \\ -\alpha_{MAX}\ M &\geq L_1 \end{aligned}$$

This implies $\alpha_{MAX} \leq min(\frac{L_2}{N}, \frac{-L_1}{M})$, and part (ii) is proved.

Proof of Theorem 7.1.4

Referring to Figure 7.1 for TOS1, if the point $(\bar{x}, y) = (0, M)$ is inside the domain of attraction, i.e. $(\bar{x}, y) = (0, M)$ satisfies $\frac{1}{C_1-(x+y)} + P_1 \leq \frac{1}{C_2-(\bar{x}+\bar{y})} + P_2$, then we have $N + M \leq C_1 - \frac{1}{(\frac{1}{C_2}+P_2-P_1)}$. This implies $T_{1,k} \leq T_{2,k}$, for every k. This ensures that path 1 is always attractive to delay-sensitive traffic, and eventually all delay-sensitive traffic will be on path 1.

Given that all delay-sensitive connections remain on path 1, we see from (6.2) and (6.3) that the condition $\frac{M}{C_2} \leq \frac{N}{C_1}$ is enough for all throughput-sensitive traffic to eventually move to path 2. This proves part (i).

Referring to Figure 7.2 for TOS2, if $(\bar{x}, y) = (0, 0)$ is inside the domain of attraction, i.e. $(\bar{x}, y) = (0, 0)$ satisfies $\frac{1}{0.5C_1 - x} + P_1 \leq \frac{1}{(C_2+0.5\bar{x})-\bar{x}} + P_2$, then we have $N \leq 0.5C_1 - \frac{1}{(\frac{1}{C_2}+P_2-P_1)}$. This implies $T_{1,k} \leq T_{2,k}$, for every k. Therefore, similar to part (i), we see that the conditions $N \leq 0.5C_1 - \frac{1}{(\frac{1}{C_2}+P_2-P_1)}$, and $\frac{M}{C_2} \leq \frac{N}{C_1}$ are sufficient for isolation. This proves part (ii).

[1] We note that $L_2 \leq N$ since otherwise we get $-M > N$, which contradicts the fact that $N, M \geq 0$. Referring to Figure 7.1, L_1 is assumed to be greater than $-M$.

Bibliography

[1] H. Ahmadi, J. Chen, and R. Guerin. Dynamic routing and call control in high-speed integrated networks. In *Workshop on Systems Engineering and Traffic Engineering, ITC'13*, pages 19–26, Copenhagen, Denmark, June 1991.

[2] C. Alaettinoglu, I. Matta, and A.U. Shankar. A scalable virtual circuit routing scheme for ATM networks. Technical Report CS-TR-3360, Department of Computer Science, University of Maryland, College Park, MD 20742, October 1994. To appear in Fourth International Conference on Computer Communications and Networks '95, Las Vegas, Nevada.

[3] C. Alaettinoğlu, A. U. Shankar K. Dussa-Zieger, and I. Matta. Design and implementation of MaRS: A routing testbed. *Journal of Internetworking: Research and Experience*, 5(1):17–41, March 1994.

[4] C. Alaettinoğlu and A. U. Shankar. Viewserver hierarchy: A new inter-domain routing protocol. In *IEEE INFOCOM '94*, Toronto, Canada, June 1994.

[5] P. Almquist. Type of service in the Internet Protocol suite. Technical Report RFC-1349, Network Working Group, July 1992.

[6] D. Anick, D. Mitra, and M. Sondhi. Stochastic theory of a data handling system with multiple sources. *Bell Syst. Tech. J.*, 61:1871–1894, 1982.

[7] G. Ash, J. Chen, A. Frey, and B. Huang. Real-time network routing in a dynamic class-of-service network. In *13th ITC*, Copenhagen, Denmark, 1991.

[8] S. Bahk and M. El Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *SIGCOMM '92*, pages 53–64, Baltimore, Maryland, August 1992.

[9] A. Ballardie, P. Francis, and J. Crowcroft. Core based trees. In *SIGCOMM '93*, San Francisco, California, September 1993.

[10] K-H. Becker and M. Dorfler. *Dynamical Systems and Fractals*. Cambridge University Press, 1989.

[11] J. Behrens and J.J. Garcia-Luna-Aceves. Distributed, scalable routing based on link-state vectors. In *ACM SIGCOMM '94*, pages 136–147, September 1994.

[12] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Inc., 1987.

[13] D.P. Bertsekas. Dynamic behavior of shortest path routing algorithms for communication networks. *IEEE Transactions on Automatic Control*, 27(1):60–74, February 1982.

[14] J. Bolot and A.U. Shankar. A discrete-time stochastic approach to flow control dynamics. In *GLOBECOM '92*, Orlando, Florida, December 1992.

[15] J-C. Bolot and A.U. Shankar. Analysis of a fluid approximation to flow control dynamics. In *IEEE INFOCOM '92*, Florence, Italy, May 1992.

[16] B. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: An overview. Internet Draft, October 1993.

[17] L. Breslau, D. Estrin, and L. Zhang. A simulation study of adaptive source routing in integrated services networks. Available by anonymous ftp at catarina.usc.edu:pub/breslau, September 1993.

[18] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. Technical Report RFC-1195, Digital Equipment Corporation, December 1990.

[19] H. Chen and A. Mandelbaum. Discrete flow networks: Diffusion approximations and bottlenecks. *Annals of Probability*, 19(4):1463–1519, October 1991.

[20] S-P. Chung, A. Kashper, and K. Ross. Computing approximate blocking probabilities for large loss networks with state-dependent routing. *IEEE/ACM Transactions on Networking*, 1(1):105–115, February 1993.

[21] S-P. Chung and K. Ross. Reduced load approximations for multirate loss networks. *IEEE Transactions on Communications*, 41(8):1222–1231, August 1993.

[22] D.D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *SIGCOMM '92*, pages 14–26, Baltimore, Maryland, August 1992.

[23] J. Cobb and M. Gouda. Flow theory: Verification of rate-reservation protocols. In *IEEE International Conference on Network Protocols '93*, San Francisco, California, October 1993.

[24] D. Comer and R. Yavatkar. FLOWS: Performance guarantees in best effort delivery systems. In *IEEE INFOCOM, Ottawa, Canada*, pages 100–109, April 1989.

[25] R.L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[26] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM '89*, pages 1–12, Austin, Texas, September 1989.

[27] D. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *CACM*, 35(6):85–98, June 1992.

[28] E. Dijkstra. A note on two problems in connection with graphs. *Numer. Math.*, 1:269–271, 1959.

[29] Z. Dziong and J. Roberts. Congestion probabilities in a circuit-switched integrated services network. *Performance Evaluation*, 7:267–284, 1987.

[30] A. Economides and J. Silvester. Optimal routing in a network with unreliable links. In *IEEE INFOCOM '88*, pages 288–297, August 1988.

[31] A. Elwalid and D. Mitra. Effective bandwidth of general markovian traffic sources and admission control of high-speed networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, June 1993.

[32] D. Ferrari. Real–time communication in packet–switching wide–area networks. Technical Report TR-89-022, International Computer Science Institute, Berkeley, California, May 1989.

[33] D. Ferrari. Client requirements for real–time communication services. *IEEE Communications Magazine*, 28(11), November 1990.

[34] D. Ferrari and D.C. Verma. Quality of service and admission control in ATM networks. Technical Report TR-90-064, International Computer Science Institute, Berkeley, California, December 1990.

[35] J. Filipiak. *Modeling and Control of Dynamic Flows in Communication Networks*. New York: Springer-Verlag, 1988.

[36] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. In *ACM SIGCOMM '93*, San Francisco, California, September 1993.

[37] F. Le Gall and J. Bernussou. An analytical formulation for grade of service determination in telephone networks. *IEEE Transactions on Communications*, COM-31(3):420–424, March 1983.

[38] M.L. Gardner, I.S. Loobeek, and S.N. Cohn. Type-of-service routing with loadsharing. In *GLOBECOM '87*, Tokyo, Japan, November 1987.

[39] M.R. Garzia and C.M. Lockhart. Nonhierarchical communications networks: An application of compartmental modeling. *IEEE Transactions on Communications*, 37:555–564, June 1989.

[40] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Addison-Wesley Publishing Company, 1990.

[41] A. Girard and M. Bell. Blocking evaluation for networks with residual capacity adaptive routing. *IEEE Transactions on Communications*, COM-37:1372–1380, 1989.

[42] D. Glazer and C. Tropper. A new metric for dynamic routing algorithms. *IEEE Transactions on Communications*, pages 360–367, March 1990.

[43] A. Greenberg and P. Wright. Design and analysis of master/slave multiprocessors. *IEEE Transactions on Computers*, 40(8):963–976, August 1991.

[44] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Select. Areas Commun.*, SAC-9(7):968–981, September 1991.

[45] S. Gupta. *Performance Modeling and Management of High-Speed Networks*. PhD thesis, University of Pennsylvania, Department of Systems, 1993.

[46] S. Gupta, K. Ross, and M. ElZarki. Routing in virtual path based ATM networks. In *GLOBECOM '92*, pages 571–575, 1992.

[47] S. Gupta, K. Ross, and M. ElZarki. On routing in ATM networks. In *IFIP TC6 Task Group/WG6.4 Workshop on Modeling and Performance Evaluation of ATM Technology*. H. Perros, G. Pujolle, and Y. Takahashi (Editors). Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1993.

[48] R.-H. Hwang. *Routing in High-Speed Networks*. PhD thesis, University of Massachusetts, Department of Computer Science, May 1993.

[49] R-H. Hwang, J. Kurose, and D. Towsley. MDP routing in ATM networks using virtual path concept. In *IEEE INFOCOM*, pages 1509–1517, Toronto, Ontario, Canada, June 1994.

[50] J. Hyman, A. Lazar, and G. Pacifici. A separation principle between scheduling and admission control for broadband switching. *IEEE J. Select. Areas Commun.*, SAC-11(4):605–616, May 1993.

[51] R. Jain and S.A. Routhier. Packet trains - measurements and a new model for computer network traffic. *IEEE JSAC*, 4(6):986–995, September 1986.

[52] D. Johnson. NSFnet report. In *19th IETF*, pages 377–382, University of Colorado, National Center for Atmospheric Research, December 1990.

[53] S. Jordan and P. Varaiya. Throughput in multiple service, multiple resource communication networks. *IEEE Transactions on Communications*, 39(8):1216–1222, August 1991.

[54] I. Kamel and C. Faloutsos. Parallel R-trees. In *ACM SIGMOD*, pages 195–204, San Diego, CA, June 1992.

[55] J. Kaufman. Blocking in a shared resource environment. *IEEE Transactions on Communications*, 29(10):1474–1481, October 1981.

[56] K. Keeton, B. Mah, S. Seshan, R. Katz, and D. Ferrari. Providing connection-oriented network services to mobile hosts. In *USENIX Symposium on Mobile and Location-Independent Computing*, Cambridge, Massachusetts, August 1993.

[57] F.P. Kelly. Blocking probabilities in large circuit-switched networks. *Adv. Appl. Prob.*, 18:473–505, 1986.

[58] S. Keshav, A. Agrawala, and S. Singh. Design and analysis of a flow control algorithm for a network of rate allocating servers. In *IFIP WG 6.1/WG 6.4 Second International Workshop on Protocols for High-Speed Networks*, pages 55–72, Palo Alto, CA, November 1990.

[59] G. Kesidis, J. Walrand, and C.-S. Chang. Effective bandwidths for multiclass markov fluids and other ATM sources. *IEEE/ACM Transactions on Networking*, 1(4):424–428, August 1993.

[60] A. Khanna and J. Zinky. A revised ARPANET routing metric. In *ACM SIGCOMM '89*, pages 45–56, September 1989.

[61] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* Brooks/Cole Publishing Company, 1991.

[62] L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delay.* New York: McGraw Hill, 1964.

[63] L. Kleinrock. *Queueing Systems*, volume I and II. New York: Wiley, 1976.

[64] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks. *Computer Networks*, 1:155–174, 1977.

[65] B. Kuo. *Automatic Control Systems.* Prentice-Hall, Inc., fourth edition, 1983.

[66] R. LaMaire, A. Krishna, and H. Ahmadi. Analysis of a wireless MAC protocol with client-server traffic. In *IEEE INFOCOM '93*, San Francisco, California, 1993.

[67] S. S. Lavenberg. *Computer Performance Modeling Handbook*. Academic Press, 1983.

[68] S. Leutenegger and X-H. Sun. Distributed computing feasibility in a non-dedicated homogeneous distributed system. In *Supercomputing '93*, November 1993.

[69] M. Litzkow, M. Livny, and M. Mutka. Condor – a hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*, San Jose, CA, June 1988.

[70] G. Louth, M. Mitzenmacher, and F.P. Kelly. Computational complexity of loss networks. *Theoretical Computer Science*, 125:45–59, 1994.

[71] W. Lovegrove, J. Hammond, and D. Tipper. Simulation methods for studying nonstationary behavior of computer networks. *IEEE J. Select. Areas Commun.*, 8(9):1696–1708, December 1990.

[72] I. Matta and A.U. Shankar. Type-of-service in adaptive next-hop routing. Technical Report CS-TR-2963, Department of Computer Science, University of Maryland, College Park, MD 20742, September 1992. Available by anonymous ftp at ftp.cs.umd.edu:pub/MaRS/Papers.

[73] I. Matta and A.U. Shankar. On the interaction between gateway scheduling and routing. Technical Report CS-TR-3102, Department of Computer Science, University of Maryland, College Park, MD 20742, July 1993. Available by anonymous ftp at ftp.cs.umd.edu:pub/MaRS/Papers.

[74] I. Matta and A.U. Shankar. An iterative approach to comprehensive performance evaluation of integrated services networks. In *IEEE International Conference on Network Protocols '94*, Boston, Massachusetts, October 1994. Extended version submitted for possible journal publication.

[75] I. Matta and A.U. Shankar. On the interaction between gateway scheduling and routing. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '94*, pages 84–88, Durham, North Carolina, January 1994.

[76] I. Matta and A.U. Shankar. Type-of-service routing in dynamic datagram networks. In *IEEE INFOCOM*, pages 992–999, Toronto, Ontario, Canada, June 1994. Extended version will appear in the *IEEE Journal on Selected Areas in Communications – Special Issue on the Internet*.

[77] I. Matta and A.U. Shankar. *Z*-iteration: A simple method for throughput estimation in time-dependent multi-class systems. In *ACM SIGMETRICS/PERFORMANCE '95*, pages 126–135, Ottawa, Canada, May 1995.

[78] J. McQuillan, I. Richer, and E. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[79] D. Mitra and J. Seery. Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks. *IEEE Transactions on Communications*, 39(1):102–116, January 1991.

[80] D. Mitra and P. Weinberger. Probabilistic models of database locking: Solutions, computational algorithms, and asymptotics. *J. ACM*, 31(4):855–878, October 1984.

[81] N. Mitrou and D. Pendarakis. Cell-level statistical multiplexing in ATM networks: Analysis, dimensioning, and call-acceptance control w.r.t. QOS criteria. In *Queueing, Performance and Control in ATM (ITC-13)*, pages 7–12. J. Cohen and C. Pack (Editors). Elsevier Science Publishers B.V. (North-Holland), 1991.

[82] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang. An architectural comparison of ST-II and RSVP. In *IEEE INFOCOM*, pages 716–725, Toronto, Ontario, Canada, June 1994.

[83] J. Moy. OSPF version 2. RFC 1247, Network Information Center, SRI International, July 1991.

[84] A. Mukherjee and J.C. Strikwerda. Analysis of dynamic congestion control protocols: A fokker-plank approach. In *ACM SIGCOMM '91*, Zurich, Switzerland, September 1991.

[85] R. Nagarajan, J. Kurose, and D. Towsley. Local allocation of end-to-end quality-of-service in high-speed networks. In *IFIP TC6 Workshop on Modelling and Performance Evaluation of ATM Technology*, page 2.2, January 1993.

[86] K. Ogata. *Discrete-Time Control Systems.* Prentice-Hall, Inc., 1987.

[87] Y. Ohba, M. Murata, and H. Miyahara. Analysis of interdeparture processes for bursty traffic in ATM networks. *IEEE J. Select. Areas Commun.*, 9(3):468–476, April 1991.

[88] S. Ohta and K. Sato. Dynamic bandwidth control of the virtual path in an asynchronous transfer mode network. *IEEE Transactions on Communications*, 40(7):1239–1247, July 1992.

[89] A. Parekh. A generalized processor sharing approach to flow control in integrated services networks. Technical Report LIDS-TR-2089, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992.

[90] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. In *ACM SIGCOMM '94*, University College London, London, UK, September 1994.

[91] M. Prycker. *Asynchronous Transfer Mode - Solution for Broadband ISDN.* Ellis Horwood, 1991.

[92] J. Roberts. A service system with heterogeneous user requirements - application to multi-services telecommunications systems. In *Performance of Data Communications Systems and Their Applications*, pages 423–431. G. Pujolle (Editor). North-Holland Publishing Company, 1981.

[93] H. Schulzrinne, J. Kurose, and D. Towsley. An evaluation of scheduling mechanisms for providing best-effort real-time communication in wide-area networks. In *IEEE INFOCOM*, pages 1352–1361, Toronto, Ontario, Canada, June 1994.

[94] A.U. Shankar, C. Alaettinoğlu, K. Dussa-Zieger, and I. Matta. Performance comparison of routing protocols under dynamic and static file transfer connections. *ACM Computer Communication Review*, October 1992.

[95] A.U. Shankar, C. Alaettinoğlu, I. Matta, and K. Dussa-Zieger. Performance comparison of routing protocols using MaRS: Distance-vector versus link-state. In *ACM SIGMETRICS/PERFORMANCE*, volume 20, pages 181–192, Newport, Rhode Island, June 1992.

[96] S. Sibal and A. DeSimone. Controlling alternate routing in general-mesh packet flow networks. In *ACM SIGCOMM '94*, pages 168–179, September 1994.

[97] H. A. Taha. *Operations Research : An Introduction.* Macmillan publishing Co., second edition, 1976.

[98] L. Takács. *Introduction to the Theory of Queues*, pages 174–187. Greenwood Press, Westport, Connecticut, 1961.

[99] H. Takagi. Queueing analysis of polling models: An update. In *Stochastic Analysis of Computer and Communication Systems*, pages 267–318. H. Takagi (Editor). Elsevier Science Publishers B.V. (North-Holland), 1990.

[100] D. Tipper and M.K. Sundareshan. Numerical methods for modeling computer networks under nonstationary conditions. *IEEE J. Select. Areas Commun.*, 8(9):1682–1695, December 1990.

[101] S. Tripathi and A. Duda. Time-dependent analysis of queueing systems. *INFOR*, 24(3):199–219, 1986.

[102] P. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. In *Proc. SIGCOMM '88*, pages 35–42, Stanford, California, 1988.

[103] D.C. Verma, H. Zhang, and D. Ferrari. Delay jitter control for real–time communication in a packet–switching network. In *IEEE TRICOMM*, pages 35–43, Chapel Hill, North Carolina, April 1991.

[104] J. Wieselthier, C. Barnhart, and A. Ephremides. Optimal admission control in circuit-switched multihop radio networks. In *31st Conference on Decision and Control*, Tucson, Arizona, December 1992.

[105] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, 1991.

[106] H. Zhang and D. Ferrari. Rate-controlled static priority queueing. Technical Report TR-92-003, International Computer Science Institute, Berkeley, California, January 1992.

[107] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *SIGCOMM '90*, pages 19–29, Philadelphia, Pennsylvania, September 1990.

[108] W. Zhu and S. Chanson. Adaptive threshold-based scheduling for real-time and non-real-time traffic. In *IEEE RTSS '92*, pages 125–135, Phoenix, Arizona, December 1992.

[109] J. Zinky. Private communication, 1992.